Reel Recommendations: Leveraging Reinforcement Learning for Tailored LetterBoxd Recommendations

Luke Benham
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`lnb6grp@virginia.edu`

Michael Fatemi
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`gsk6me@virginia.edu`

Kasra Lekan
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`kl5sq@virginia.edu`

Abstract

We use offline reinforcement learning and language modeling to provide movie recommendations based on user data from LetterBoxd, popular movie review and cataloging website, and movie data from TMDB. We implement DeepFM, a framework for converting sparse features to dense embeddings. Additionally, we formulate the recommendation task as a language modeling task by fine-tuning GPT-2 to predict movie ratings based on a sequence of previously rated movies. Our implementation can be found at `https://github.com/anrath/LetterBoxd-RL-Rec`.

1 Introduction

Letterboxd is a popular movie review and cataloging website among cinephiles (similar to GoodReads for books). The large number of reviews, the user social graph, and the long-form text reviews that people leave for their favorite movies could facilitate an effective recommendation system. Because we have an existing dataset, our learning algorithms will operate fully offline.

We use natural language descriptions of users and movies and embed them using SentenceTransformers. We will also include explicit feature vectors for each movie (such as genre and length).

We considered methods that (approximately) learned user embeddings and those that learned movie embeddings. DeepFM is a blend between these two but primarily focuses on movie embeddings while a fine-tuned GPT-2 model solely learns movie embeddings (through natural language formulations of movie data).

2 Related Work

Afsar et. Al 2022 researches RL recommender algorithms which are more similar to our final goal. The Feng et. Al paper introduces a new type of RL recommender system that models the user-item interaction as the primary data item and outcome. Quisar 2020 uses Long Short-Term Memory as its algorithm for text classification in movie reviews which could be a useful technique for us to use on text-based features. Naeem 2022 and Mehra et. Al 2018 both implement a variety of classification algorithms and find that a Support Vector Machine is generally the most accurate. Other clustering techniques are implemented by Aditya et. Al 2018 shows the benefits of k-means clustering.

2.1 Reinforcement Learning based Recommender Systems: A Survey.

M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement Learning based Recommender Systems: A Survey. ACM Comput. Surv. 55, 7, Article 145 (July 2023), 38 pages. https://doi.org/10.1145/3543846

The paper "Reinforcement Learning based Recommender Systems: A Survey" by Afsar, Crump, and Far provides a comprehensive overview of the use of reinforcement learning (RL) and deep reinforcement learning (DRL) in recommender systems (RSs). It highlights the shift towards formulating recommendation as a sequential decision problem, better capturing the dynamic user-system interaction. By reviewing RL and DRL-based methods, the paper proposes a framework for RL-based recommender systems with four components: state representation, policy optimization, reward formulation, and environment building. It discusses emerging topics and important trends.

2.2 Deep reinforcement learning based recommendation with explicit user-item interactions modeling.

Liu, Feng, et al. "Deep reinforcement learning based recommendation with explicit user-item interactions modeling." arXiv preprint arXiv:1810.12027 (2018). https://doi.org/10.48550/arXiv.1810.12027

The paper "Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling" introduces a novel recommendation framework utilizing deep reinforcement learning (DRR) to enhance recommendation systems by explicitly modeling user-item interactions. It employs an "Actor-Critic" mechanism for capturing the dynamic nature of user preferences and the sequential decision-making process inherent in recommendations. By incorporating user-item interaction data directly into the model, the proposed DRR framework aims to provide more personalized and effective recommendations. The framework is evaluated through extensive experiments on real-world datasets, demonstrating superior performance over existing state-of-the-art methods.

2.3 Sentiment analysis of IMDb movie reviews using long short-term memory.

Qaisar, S. M. (2020, October). Sentiment analysis of IMDb movie reviews using long short-term memory. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-4). IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9257657

This paper introduces a sentiment analysis model for IMDb movie reviews using Long Short-Term Memory (LSTM), a type of Recurrent Neural Network. The study preprocesses and partitions a dataset of 50,000 reviews for training and testing, achieving a classification accuracy of 89.9%. It underscores the effectiveness of LSTM in text sentiment analysis, suggesting potential for further accuracy improvements through data preprocessing and the use of ensemble classifiers or deep learning approaches.

2.4 Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms.

Naeem, Muhammad Zaid, et al. "Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms." PeerJ Computer Science 8 (2022): e914. https://peerj.com/articles/cs-914/

The paper "Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms" by Naeem et al. investigates the application of various machine learning models, including SVM, Naive Bayes, Random Forest, and Gradient Boosting classifiers, for sentiment analysis on IMDb movie reviews. It emphasizes preprocessing techniques and feature extraction methods like TF-IDF and BoW to enhance classification accuracy. The study highlights the effectiveness of SVM with TF-IDF features achieving the highest accuracy and discusses the impact of preprocessing and feature engineering on sentiment classification performance.

2.5 Sentiment analysis of user entered text.

Mehra, Sourav, and Tanupriya Choudhury. "Sentiment analysis of user entered text." 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS). IEEE, 2018. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8769136

The paper "Sentiment Analysis of User Entered Text" by Sourav Mehra and Tanupriya Choudhury focuses on comparing the performance of SVM (Support Vector Machine) and Naïve Bayes classifiers for sentiment analysis of movie reviews. Utilizing datasets from IMDb and Cornell University, the study assesses classification accuracies across different dataset sizes. The findings demonstrate SVM's slight superiority over Naive Bayes in accuracy, suggesting the potential for improved classification outcomes with larger training datasets. This research underlines the importance of selecting appropriate machine learning algorithms and dataset sizes in sentiment analysis tasks.

2.6 Comparative analysis of clustering techniques for movie recommendation.

Aditya, T. S., Karthik Rajaraman, and M. Monica Subashini. "Comparative analysis of clustering techniques for movie recommendation." MATEC Web of Conferences. Vol. 225. EDP Sciences, 2018. https://www.matec-conferences.org/articles/matecconf/pdf/2018/84/matecconf_ses2018_02004.pdf

The paper "Comparative Analysis of Clustering Techniques for Movie Recommendation" explores four clustering algorithms: K-means, Agglomerative, Birch, and Mean-shift, applied to the TMDB 5000 Movie Dataset. It aims to identify the most effective technique for movie recommendation by examining how different algorithms and feature selections impact the clustering results. The study finds K-means and Birch to be particularly effective, highlighting the importance of feature selection and the number of clusters in achieving high-quality recommendations. It underscores the practical utility of clustering in unsupervised learning for personalized movie recommendations.

3 Preliminaries

Overview. We use the DeepFM framework, which allows for efficient use of feature vectors. We create "user vectors" and "movie vectors", which jointly are used as a context to predict a reward value, $\hat{R}$. Because there is no intrinsic user information present in the dataset, we must use information about their past interactions with movies to estimate their future preferences. For now, we encode this by making user vectors trainable parameters (i.e., when creating the loss function for $\hat{R}$, the user vector can be optimized to represent learning their preferences).

Summary of the DeepFM Framework. DeepFM uses sparse features (such as movie and user metadata) and constructs "dense features" from those. DeepFM can be conceptually viewed as an extension of Linear Contextualized Bandits, but due to additional complexity, it is optimized with stochastic gradient descent rather than direct linear regression. Several factors are linearly combined to generate a final output logit, which can represent a rating prediction or an action. The input context is a user vector concatenated with a movie vector, and the action is represented as a single output logit, representing whether the movie should be recommended to the user.

The context vector is projected into a set of low-dimensional "dense features". These dense features are then converted into a compatibility vector by taking pairwise inner products between them to measure similarity. The results from these inner products are added to the final output logit. Additionally, these dense features are passed to a shallow multi-layer perceptron, which allows for additional tuning of the output logit.

DeepFM can be viewed as a sum of a linear contextual bandit, via the linear projection from sparse features to output logit, a factorization machine, which is the set of pair-wise inner products between dense features, and a multi-layer perceptron, stemming from the dense features. Therefore, it is highly versatile and adaptive to the specific problem at hand. Additionally, each of these components can be nullified by using zero weights, and thus DeepFM can collapse to any of these individual components if demonstrated to be the optimal modeling approach.

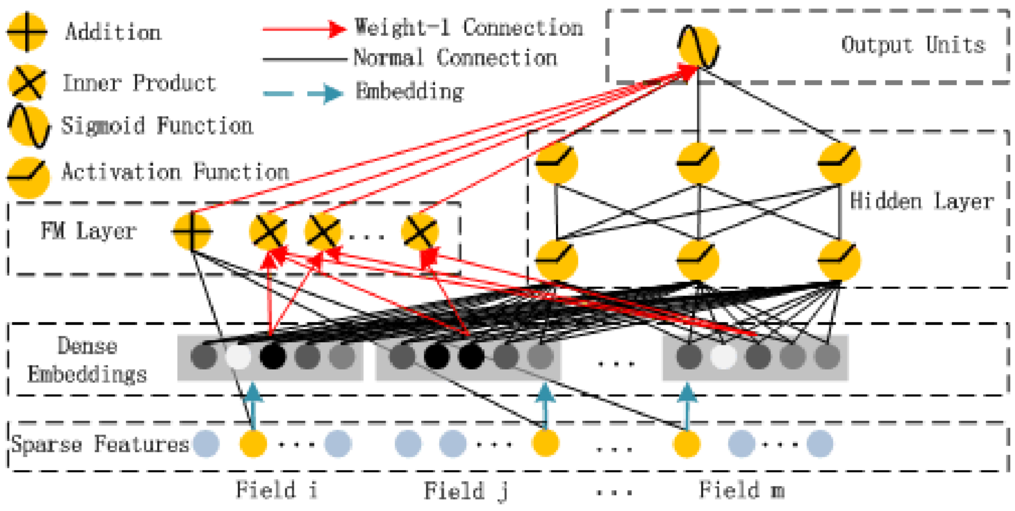

Figure 1: "Wide & deep architecture of DeepFM" from the original paper.

4 Methods

4.1 Data

We wrote Python scripts using BeautifulSoup to scrape the user pages for roughly 6500 top LetterBoxd users to extract their movie-watching data. This included timestamps, a 5-star rating, and boolean values for whether a user liked, rewatched, or reviewed the movie. We scraped the text of these user reviews as well.

We downloaded an initial Kaggle dataset of movie data up to 2022 and supplemented it with data we gathered from the TMDB API for movies released from 2022-2024. These dataset fields are listed in the Appendix.

We featurized the movie data relevant to our recommendation algorithms. For categorical variables like `genres`, `production_countries`, and `spoken_languages`, we used one-hot encoding. We converted time-based values like `release_date` and `year_released` to ordinal values. Additionally, we encoded `overview`, a natural language description of the movie, using the pre-trained model all-MiniLM-L6-v2 from the SentenceTransformers package. We chose this model because of its high performance on sentence embedding tasks and fast embedding speed.

4.2 Learning and Recommendations

DeepFM Learning Objective. Here we apply the DeepFM framework. For a given user/movie pair, we assign a reward of $r = 1$ when the user gave the movie a rating ≥ 7, and a reward of $r = 0$ otherwise. We then train our model to regress the reward value through mean-squared-error loss. To prevent overestimating model performance, we create a baseline predictor which simply estimates $r = 1$ when a movie's average rating (across all users) is ≥ 7, and $r = 0$ otherwise. Thus, if our model can outperform the baseline, it has learned a method of recommending movies to users in a more selective/personalized fashion than global metrics for movie quality.

Language Model Learning Objective. We also attempted to use movie descriptions directly as inputs into an autoregressive language model: GPT-2. The motivation was that the pre-trained representations from GPT-2 would be powerful, and by using an end-to-end approach with self-attention, we could avoid the step of training “user vectors" and instead directly infer user preferences from their interaction history.

Under this approach, we treat movie recommendations as a sequence modeling objective. Let y_t and x_t represent the user's rating and the information about the movie at time t. We give the model a reward proportional to the rating the user provides, and then train the model to estimate the reward:

$$R_t \sim P(y_t \mid y_{t-1}, x_{t-1}, \ldots, y_0, x_0)$$

We let N represent the "context length": the number of ratings the language model can view in order to make its prediction. For brevity, we will from now on denote the sequence of movies and user feedback as τ_N (representing "trajectory"). We define loss via a cross-entropy objective. Because we assume implicit variance in user ratings, model the target distribution, $P(y_{N+1} \mid \tau_N)$, as a softer version of categorical classification. Given y_{N+1}, we train the model with cross-entropy loss against the distribution:

$$P(y \mid \tau_N) = \begin{cases} 0.25 & \text{if } y_{N+1} = y + 1 \\ 0.5 & \text{if } y_{N+1} = y \\ 0.25 & \text{if } y_{N+1} = y - 1 \end{cases}$$

If $y_{N+1} \in \{0, 10\}$, we clip $y = -1$ and $y = 11$ and rescale the remaining probabilities to sum to 1. The purpose of this is to enforce some numerical structure: The model should be penalized less harshly for similar predictions than for distant ones.

5 Experiments

5.1 DeepFM Experiments

First, we created a small development set consisting of 30 unique movies (and a large set of users who have provided ratings for those movies). We found that after training for a small number of epochs (50), our model predicted reward correctly with 81% accuracy, where "correctness" is defined as $\frac{\mathbb{I}\{\text{Round}(\hat{R})=r\}}{N}$ (N being the number of instances in the training set). The baseline estimator, using no user personalization, had an accuracy of 70%.

Having implemented DeepFM for our recommendations, we scaled up our experiment to test various dataset sizes (results shown in Table 1). We noted that our model strongly overfitted to the training dataset.

Table 1: Accuracy of DeepFM before overfitting correction.

Split	Acc. (No Personalization)	100 Epoch Acc. (train)	200 Epoch Acc. (train)	100 Epoch Acc. (test)	200 Epoch Acc. (test)
30 train, 0 test	0.7106	0.9040	0.9889	-	-
500 train, 100 test	0.769	0.7860	0.8430	0.5695	0.5732
1000, 150 test	0.745	0.6030	0.9040	0.5170	0.5369
9000 train, 1000 test	0.7322	0.8437	0.8878	0.5641	0.5662

5.2 DeepFM Overfitting

Accuracy We found that the DeepFM model was leading to overfitting for the recommendations because the training accuracy would climb to over 90% but the testing accuracy remained between 50% and 60%. This was especially true of the smallest datasets used in development which approached 99% accuracy on the training set which made it obvious that other techniques would be necessary to limit the overfitting issue.

Dropout The primary adaptations for the overfitting issue were the implementations of the dropout and L2 regularization. The dropout was applied to the DeepFM model where it contains the variable p that is the probability for a given element to be zeroed. When $p = 0.5$ then with a probability of 50% the element will be zeroed which ensures that the model does not learn too specifically the training data but instead generalizes.

L2 Regularization was implemented in the Adam optimizer. By adding a very small weight_decay term to the optimizer on the order of $1e^{-5}$ the optimizer adds a regularization term to the loss function proportional to the squares of the weights of the model. This has the effect of shrinking the parameter estimates and biases the model towards simpler solutions. By punishing complex weights, L2 regularization discourages training set memorization and limits overfitting.

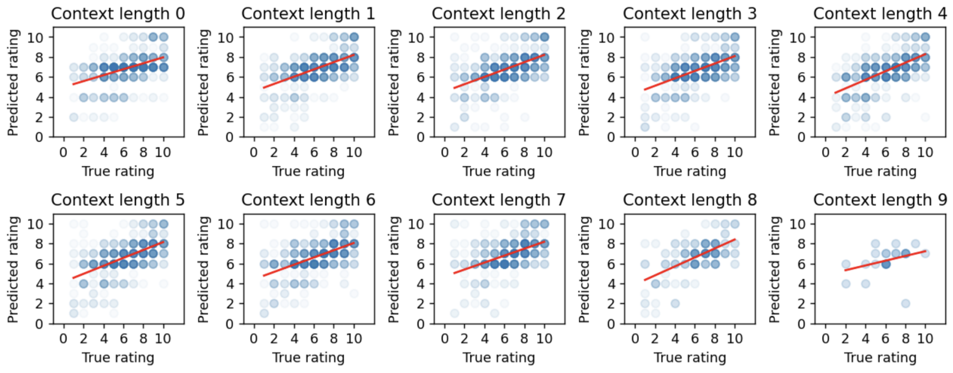

Figure 2: Scatterplot (Unweighted)

5.3 Language Model Experiments

Given our large dataset of movie rating histories, sorted chronologically and collated by user, we uniformly sample sliding context windows τ_N. For each movie that the user rates in a given context window, we format a context string for each movie that includes the movie's description, the genre, the languages spoken, and the average vote. Then, we add the string `Rating: X / 10`, where `X` is replaced with the rating value.

We sample context trajectories of length 10, and truncate them to fit in GPT-2's context window (1024). Improved sampling techniques to reduce skewed sampling towards users with disproportionately more ratings than others can be left to future work. We then train the model to predict all movie ratings simultaneously, with a decoder-only attention mask to enforce rating predictions to only be made based on prior ones. We experiment with two loss objectives:

1. **Unweighted**: Apply an autoregressive loss, in which each token from the formatted movie string is trained to predict the next token. This means the model is additionally trained to predict movie details, such as the description and genre.
2. **Weighted**: Here, we only apply the loss function to rating tokens. The motivation is that less of the training objective would be dedicated to memorizing movie details, in case such information bottlenecked the model's ability to predict ratings correctly.

We train each model with $100,000$ rating trajectories, or a total of approximately $100,000,000$ tokens. This takes approximately 4 hours on a single NVIDIA A100 GPU (which we accessed through Rivanna). We use the Adam optimizer and a learning rate of 10^{-4}.

5.4 Language Model Results

We evaluate the model's performance by randomly sampling $n = 1000$ rating trajectories from a test split of the dataset. Then, we compare the model's predicted rating to the user's true rating. We plot these in 5. We find that generally, model performance increases as context length increases. It is difficult to determine whether these benefits are from the language model/LM knowing more about the user given their rating history, or from the general theme of LMs achieving better performance on a task after being shown a few examples. The weighted model has slightly higher recall, and the unweighted model has slightly higher precision, indicating that it is ambiguous whether changing the loss objective had a strong positive impact.

We show scatterplots as a function of context length for each model: The model trained with the unweighted objective is shown in 2, and the model trained with the weighted objective is shown in 3.

We find that the models are able to obtain the following average rewards: As is shown in 5.4, the language models under both training objectives are able to achieve rewards of approximately 0.73. This is similar to the reward on the training set that DeepFM was able to receive, but surprisingly, GPT-2 is able to generalize much more effectively. We hypothesize that this is due to GPT-2's large-scale pretraining.

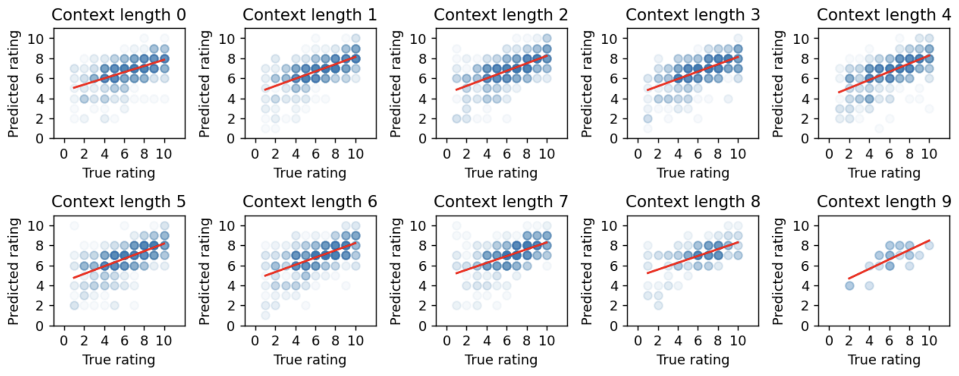

Figure 3: Scatterplot (Weighted)

Context Length	Weighted	Unweighted
0	0.7087	0.6533
1	0.7411	0.6907
2	0.7197	0.7099
3	0.7225	0.6946
4	0.7331	0.7248
5	0.7334	0.7240
6	0.7237	0.7381
7	0.7042	0.7353
8	0.6545	0.7068
9	0.6800	0.8000

6 Conclusions

We evaluate methods for offline movie recommendation using a dataset from LetterBoxd's top users. We find that DeepFM works well on training data but falters due to overfitting when evaluated on the test data. We attempt various standard methods of correcting overfitting, but ultimately are unable to correct entirely for the overfitting issue. We also attempt to formulate the recommendation task as a language modeling token prediction of the movie score given a set of 10 previous movies watched. This approach has several benefits including the use of generalized representations of natural language data for each movie as well as the ability to generalize across users. We also include a file allowing users to test the recommendation in a pseudo-online setting by inputting a list of movies and requesting a recommendation. Future work on this recommendation task could extend our approach to a truly online setting or use additional sources of user data like review text and the user social graph.

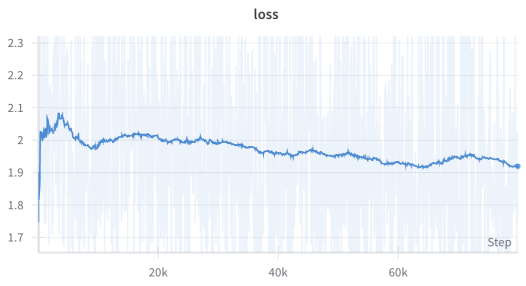

Figure 4: Learning Curve for GPT-2 Model

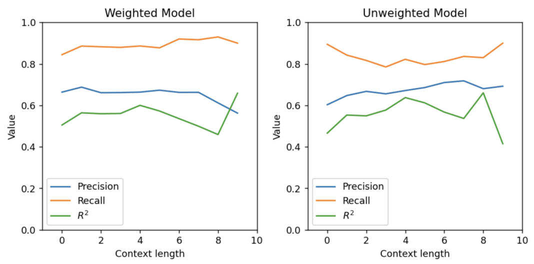

Figure 5: Comparison between Weighted and Unweighted Models

References

[1] Sarker KU, Saqib M, Hasan R, Mahmood S, Hussain S, Abbas A, Deraman A. A Ranking Learning Model by K-Means Clustering Technique for Web Scraped Movie Data. Computers. 2022; 11(11):158. https://doi.org/10.3390/computers11110158

[2] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement Learning based Recommender Systems: A Survey. ACM Comput. Surv. 55, 7, Article 145 (July 2023), 38 pages. https://doi.org/10.1145/3543846

[3] Liu, Feng, et al. "Deep reinforcement learning based recommendation with explicit user-item interactions modeling." arXiv preprint arXiv:1810.12027 (2018). https://doi.org/10.48550/arXiv.1810.12027

[4] Qaisar, S. M. (2020, October). Sentiment analysis of IMDb movie reviews using long short-term memory. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-4). IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9257657

[5] Naeem, Muhammad Zaid, et al. "Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms." PeerJ Computer Science 8 (2022): e914. https://peerj.com/articles/cs-914/

[6] Mehra, Sourav, and Tanupriya Choudhury. "Sentiment analysis of user entered text." 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS). IEEE, 2018. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8769136

[7] Aditya, T. S., Karthik Rajaraman, and M. Monica Subashini. "Comparative analysis of clustering techniques for movie recommendation." MATEC Web of Conferences. Vol. 225. EDP Sciences, 2018. https://www.matec-conferences.org/articles/matecconf/pdf/2018/84/matecconf_ses2018_02004.pdf

[8] Huifeng Guo and Ruiming Tang and Yunming Ye and Zhenguo Li and Xiuqiang He. "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction." 2017 https://arxiv.org/pdf/1703.04247

Appendix

Movie dataset fields:

`_id`, `genres`, `image_url`, `imdb_id`, `imdb_link`, `movie_id`, `movie_title`, `original_language`, `overview`, `popularity`, `production_countries`, `release_date`, `runtime`, `spoken_languages`, `tmdb_id`, `tmdb_link`, `vote_average`, `vote_count`, `year_released`.