Temporally-aware Factorization Methods in Film Recommendation from Letterboxd.com Reviews

Julian Bixler
University of California, Berkeley
Berkeley, CA, USA
jbixler@berkeley.edu

Figure 1. Letterboxd.com logo, 2024[7]

Abstract

In the study of recommendation systems, datasets composed of film viewing histories have played a major role in shaping various methods as well as brought significant interest to the field. Letterboxd.com is a social media website based on tracking, saving, and sharing films with other users. The primary form of interaction are user-film interactions: with any given film, a user can mark it as watched/rewatched, rate it, review it, and add it to a list. Additionally, Letterboxd.com has a “diary” feature, where users can “log” a film they have watched on a particular date and include an accompanying review and rating. Since Letterboxd.com is also a social media site, there are also interactions between users, where they can like each others’ reviews and lists, and follow one another. This enables a dataset taken from Letterboxd.com to comprise a wealth of distinct data which can inform various feature-based recommender systems.

1 Data & Analysis

The dataset of study is an original one wherein each row represents an instance where a user has logged a film. A separate dataset containing Letterboxd-specific data (e.g. the total number of users who liked/watched/added to a list and the bins of a rating histogram) as well as film-specific data from The Movie Database (TMDb) [6], such as language, country of production, production companies, and cast and crew, is merged with this primary dataset. Table 2 in the appendix provides a detailed overview of this dataset. All code and data are publicly available at https://github.com/jbixler24/LetterboxdRecommender.

1.1 Dataset Summary

Data is scraped from all users listed in the Letterboxd’s “Popular reviews” section with public diary pages [8]. Of these 7,705 users, up to the 500 most recent reviews for each is gathered from the “diary” section on their profile page as of the week of Nov. 17th, 2024. The resulting dataset includes interactions between these users and 160,518 films, for an average of 22 entries per film.

The data include explicit feedback (ratings and likes), textual data (reviews and review tags), temporal data (review date), and sequential data, by the nature of the dataset’s construction. New features can be extracted from each of these to inform various feature-based models as well as models specific to particular types of data (e.g. sequential recommender systems). The dataset for all analysis is restricted to entries made between Oct. 1st, 2011, when Letterboxd.com was founded, to Nov. 24th, 2024.

1.2 Exploratory Data Analysis

Due to the originality of this dataset, the primary purpose of exploring the data is to evaluate the generalizability of this data to all Letterboxd users, as well as to uncover relationships between features which could be used to train models.

Figure 2 shows the distribution of all ratings, differentiated by whether the user marked the film as "liked". While films with higher ratings appear more than those with lower ratings, users are more likely to rate films on the higher end of ratings, and are even more likely to mark them as "liked". This suggests that past ratings and likes could be very good indicators of whether or not a user will watch a given film. Coupled with this fact that people are more likely to watch films they end up liking, films with higher ratings may predict whether a user will end up watching it.

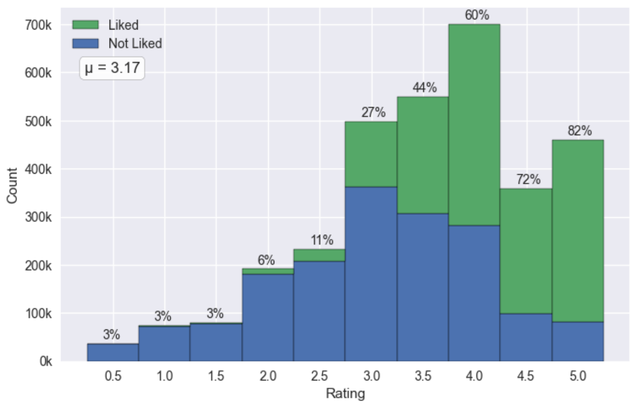

Figure 2. Rating distribution by like percentage

The co-occurrence graph in Figure 3 shows the relationships between the genres filmed can be labeled as. Each node represents a genre, with the size of the node proportional to its frequency. The size and color of the edges are proportional to the frequency with which two genres co-occur. Trends emerge between genres which can be expected to co-occur together frequently and those which very rarely do so. Once such example are the major relationships between the genres romance, comedy, and drama. We can interpret this to mean that each genre co-occurs with one another often, but also that films labeled as all three genres may occur more often than other genre triplets.

A significant part of the dataset is the temporal aspect. Not only does this enable sequential recommendation, but the review date allows seasonal, weekly, and yearly trends to become evident.

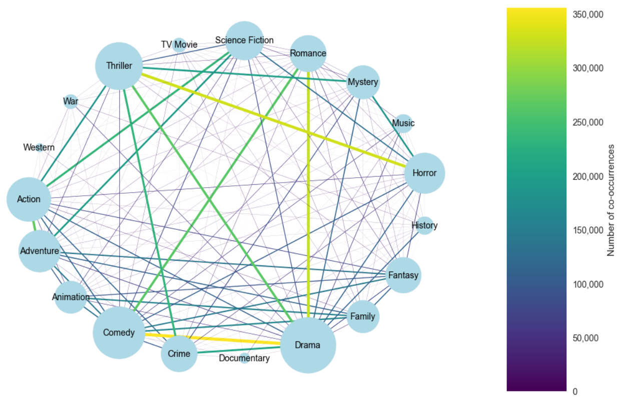

Figure 3. Genre co-occurrence network among all films logged

In Figure 4, some of the temporal trends among the rating and "like" features are apparent, from 2021 to the most recent entries in the dataset in the format of a rolling average. Both of these features appear to be very closely correlated. From 2022 onward, it is also apparent that both features experience a spike at the start of the year and a major decrease nearly three quarters into the year. Intermittent spikes throughout the year are likely due to the release of popular or well-liked films. For instance, a major spike occurs in 2023 at approximately the same time which the films *Barbie* and *Oppenheimer* were released in the United States.

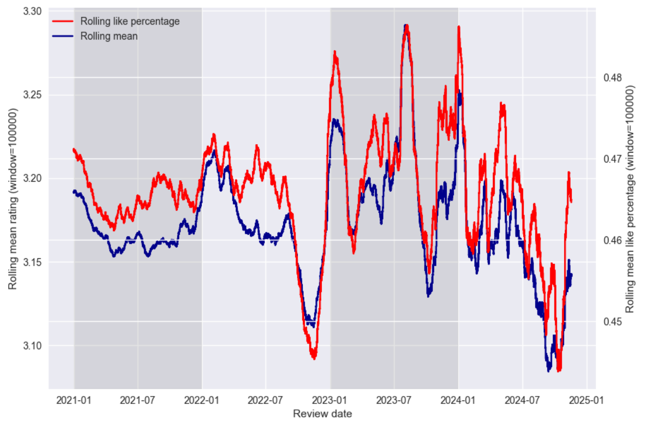

Figure 4. Temporal comparison of mean rating and like percentage

2 Predictive Task

The primary task with this Letterboxd.com dataset is to recommend films to users based on their viewing history as a way to predict which films they might next watch. In particular, this would be considered a "ranking" recommendation task.

2.1 Evaluation Metrics

Models are evaluated on the metrics log loss, Area Under ROC Curve (AUC), Precision@10 (P@10), Recall@10 (R@10), and Normalized Discounted Cumulative Gain @10 (NDCG@10/N@10).

AUC is a pair-wise metric which compares which of two films is the relevant one. An AUC of 1 is equivalent to always guessing correctly, while an AUC of 0.5 is equivalent to guessing at random.

The point-wise metrics of Precision, Recall, and NDCG measure the individual relevance of each item in the top K recommendations, where here $K = 10$. These are particularly relevant when returning a fixed number of recommendations, as is the case with the predictive task here. Precision measures the average proportion of the top K recommendations the user has watched, and Recall measures the average proportion of the top K recommendations to all films the user has watched. NDCG measures the extent to which films watched by the user in the validation or testing set appear high up in the top K recommendations [10].

The following equations (1) through (3) show the calculation for each of these point-wise metrics.

$$P@10 = \frac{1}{|U|} \sum_{u \in U} \frac{|\{i \in I_u | rank(i) \leq K\}|}{10} \quad (1)$$

$$R@10 = \frac{1}{|U|} \sum_{u \in U} \frac{|\{i \in I_u | rank(i) \leq K\}|}{|I_u|} \quad (2)$$

$$NDCG@10 = \frac{\sum_{i=1}^{10} \frac{2^{rel_i} - 1}{\log_2 i+1}}{\sum_{i=1}^{10} \frac{2^{rel_{\sigma(i)}} - 1}{\log_2 i+1}} \quad (3)$$

where $rank(i)$ denotes the position of a film was ranked for a user u.

Bayesian Personalized Ranking (BPR) is used as a baseline explicit-only feedback model to compare the selected models.

2.2 Feature Engineering

For simplicity in creating the same training and evaluation sets for each final model, minimal feature engineering is performed to the data.

Separate features are collected based on users and films to form two dataframes where each holds data for each user and film, respectively. This data is subsequently joined with the interactions from the original modeling subset to form the features used to train the FM and DeepFM models. For the user dataframe, average ratings and like percentages are calculated for each portion of the training dataset, and the same operation is performed for the film dataframe. One hot encodings of genres and of the top 10 most prevalent primary languages of the films are taken, and a percentage of each is calculated for each user.

The user and film ratings and like percentages are further transformed through a weighting with respect to their recency such that for every 10 films, its impact on the user's average rating or like percentage decreases by half. The formula for the weighted average rating for a user is provided in equation (4).

$$\text{weighted rating}(u) = \frac{\sum_{i \in I_u} r(u,i) \cdot 2^{-\frac{t_i}{10}}}{\sum_{i \in I_u} 2^{-\frac{t_i}{10}}} \quad (4)$$

The user and film features both depict related information which the FM and DeepFM can learn to associate with one another. One relation of note is the percent of each genre watched by a user and the one-hot encoding of the genres for a given film. For these features, the model learns that a user prefers certain genres over others, and that certain genres co-occur with others more frequently for some users over others.

3 Model Description

3.1 Model Architecture

The presence of textual, temporal/sequential, and explicit feedback means the data is suited toward recommendation informed by user and film features. The two primary architectures which suit this recommendation task are the factorization machine (FM) [11] and the deep factorization machine (DeepFM), both of which utilize implicit feedback as well as engineered user and item features.

The factorization machine is first described by Rendle in 2010 as a collaborative filtering model based on the Support Vector Machine (SVM) architecture [11]. The construction of FM permits solving the model equation in linear time as opposed to storing support vectors as is necessary with SVM. DeepFM is an approach which combines FM with a deep neural network, first described by Guo et. al in 2017. The original paper's authors describe DeepFM as not needing expertise feature engineering, rather simply needing raw features. [4]

An improvement of the FM model over matrix factorization and SVM techniques are its ability to use explicit feature information to capture trends which cannot be visible from interactions alone. However, the success of a FM implementation can largely be determined from how feature engineering is performed.

3.2 Cross-Validation Approach

Due to the presence of temporal features, a 5-fold time series split is constructed. The original dataset is restricted only to entries made on Jan. 1st, 2022 and onward due to the significant change in site membership following the COVID-19 pandemic [1] and the temporal trends as described in Figure 4 only becoming present from then onward. This results in a dataset of approximately 3 million entries between 7,599 users and approximately 151 thousand films.

For each fold, the model's performance is evaluated with a set of regularization parameters on the metrics from section 2. The parameter which performs the best across the metrics for all validation folds is chosen for final evaluation of the test set. These parameters range from 0.0001 to 1, increasing in each step by a factor of 10, alongside the lack of a regularizer.

The test set is constructed from the final 10% of the data. Figure 5 depicts the training/validation/testing set splits for each fold.

Due to time constraints, evaluation is performed once after the completion of each training fold to determine the regularization parameter to use on testing data. Evaluation is performed on at most 2,048 users in each validation set and the training set.

3.3 Model Implementation

All models are implemented with the LibRecommender library, whose models used here are built with Tensorflow

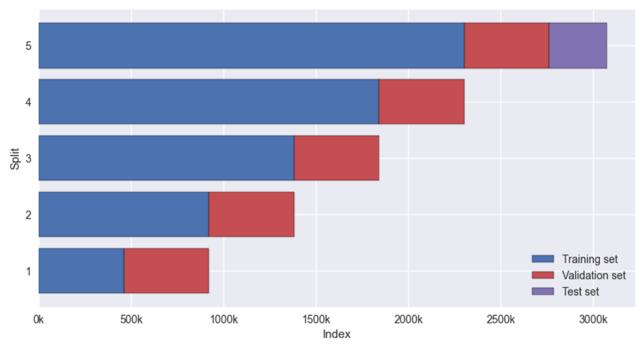

Figure 5. Training/validation/testing data split

and optimized with the Adam optimizer. In addition to being implemented with the regularization parameters described in the previous subsection, the FM model is fitted with one negative sample per true interaction.

Due to an issue with the LibRecommender library being unable to recognize users in the validation set who are not present in the training set, all cold-start interactions had to be removed from the validation set in order for training to not cause errorring. This in turn significantly reduces the generalizability of the model in cold-start recommendation, since it is not used during evaluation.

Furthermore, the resources and time needed to train each model means that there is little room for further optimization (such as regularization optimization). This additionally limits the potential for each final version of the two models.

4 Related Literature

4.1 Research in Film Recommendation

One group of datasets frequently used in the literature of recommender systems are those produced by the MovieLens recommendation system. Originally created by researchers at the University of Minnesota in 1997, MovieLens presents users with films which they can rate and tag. Since its creation, it has served alongside along longstanding datasets to inform the efficiency and accuracy of novel systems. Furthermore, like the Letterboxd dataset used here, it provides timestamps of ratings which can be used in temporal and sequential recommenders.

A class of temporally-informed recommendation models which has emerged in the past year are State Space Model-based (SSM) sequential recommenders. Originally devised for a variety of natural language tasks, SSMs describe the recommender system through differential equations. Mamba4Rec [9] and TiM4Rec [3] are two recent models which have demonstrated better performance on the MovieLens datasets, among others, while adhering to more efficient recommendation than most deep learning models.

At Netflix, deep learning methods are used which are sequentially and temporally aware. In an article published by researchers at the company, they described how continuous time context, or the exact timestamp when a user began to watch a show or film, was by far the greatest predictor in ranking tasks [12]. As a consequence of using deep learning models, they found that it was much more difficult to explain why certain recommendations were made over others.

In addition to this problem, while models can perform well on metrics in research papers, a more pressing concern is their applicability to real users of film services and advertisement. Oftentimes, how well a system may perform for a given metric is heavily dependent on how content is generated for a user on a given platform. Changes in how that content is generated, as well as the initial recommendations that are made significantly influence the user's future decisions when interacting with that platform [2]. To this extent, analysis has been performed on the usability of these datasets and the effects of feedback loops on film popularity in deployed recommender systems [12].

In this dataset, a serious consideration would not necessarily be the film recommendation process of Letterboxd, but of the respective streaming services used by the userbase. Since it may vary entirely from user to user, the Letterboxd dataset may not be the most accurate basis with which to compare various recommender systems such as FM or DeepFM.

5 Results

The results of the best-performing evaluation by regularizer on each split of cross-validation, as well as on the training set, are displayed in Table 1. In other words, if 1e-3 was a better regularizer for BPR with respect to most metrics in the second split, then the metrics for a regularizer of 1e-3 are provided in the table. The bolded values are the greatest for a particular metric in its respective split or test set across all models.

Surprisingly, the baseline BPR performed extraordinarily well compared to FM and DeepFM, doing as well half of the time as the other two models for nearly every metric. Notably, it exceeded in the point-wise metrics, while FM and DeepFM performed better on log loss and AUC.

One possibility as to the results is that features were not engineered well enough for FM to perform well. It is also possible that not enough regularization parameters or other parameters were explored for DeepFM, which could be the reason for its relative poor performance on the point-wise metrics.

Should more time be available, an improved model would make use of additional temporal features, such as the month and day of the week to capture seasonal and weekly trends. There is also enough textual data that emotional sentiments could be evaluated on film descriptions, and for each user parameters could be constructed representing their ideal proportion of various emotions in a film.

Split	Metric	BPR	FM	DeepFM
1	Log Loss	0.6901	**0.4882**	0.9545
	AUC	0.7159	**0.8652**	0.8081
	P@10	0.2007	0.1868	**0.2011**
	R@10	**0.0382**	0.0355	0.0378
	N@10	**0.5295**	0.5104	0.4825
2	Log Loss	0.6899	0.4236	0.4074
	AUC	0.7081	0.8975	**0.9010**
	P@10	**0.1500**	0.1355	0.1377
	R@10	0.0333	0.0316	**0.0350**
	N@10	**0.5230**	0.4916	0.4055
3	Log Loss	**0.6879**	0.3939	0.6712
	AUC	0.6818	**0.9142**	0.8536
	P@10	**0.2733**	0.2383	0.2027
	R@10	**0.0677**	0.0577	0.0465
	N@10	**0.5579**	0.5478	0.4768
4	Log Loss	0.6904	**0.4029**	0.4801
	AUC	0.7005	**0.9076**	0.8993
	P@10	**0.1410**	0.1163	0.1297
	R@10	**0.0355**	0.0308	0.0321
	N@10	**0.4255**	0.3706	0.4061
5	Log Loss	0.6905	0.4004	**0.4151**
	AUC	0.6979	0.9055	**0.9095**
	P@10	**0.1520**	0.1005	0.1401
	R@10	**0.0365**	0.0250	0.0334
	N@10	**0.4553**	0.3318	0.4182
Test	Log Loss	0.6922	0.3978	**0.5263**
	AUC	0.6594	**0.9102**	0.8733
	P@10	0.0236	**0.0511**	0.0249
	R@10	0.0088	0.0065	**0.0089**
	N@10	**0.1075**	0.0994	0.1065

Table 1. Performance metrics on validation splits & testing data.

5.1 Limitations

A significant limitation of the data is its generalizability to users outside of the training set. The models are built on interactions between 7,599 and approximately 151 thousand films. Since Letterboxd hosts approximately 1 million films [5] and 14 million users [1], this results in models very representative of Letterboxd's film catalog but not of its userbase.

Consequently, cold-start recommendation would necessarily be employed for the vast majority of users should this system be deployed.

Similarly, the issue arises of recommending films that did not appear in the training set. As many of the popular users of Letterboxd are likely to watch recently-released (and therefore more often discussed) films, the training data may not necessarily include the same films. In the event of an extremely popular film, it would not be able to account for large amounts of the userbase consuming it. This phenomenon is evident in the performance metrics on test data in Table 1. As the test set is constructed long after the final entry of the training set, many films have been released which are unavailable in training. As a result, we see the metrics drop significantly compared to the older validation data.

Acknowledgments

To my wonderful girlfriend Ysabelle, for supporting me through all the nights I put into scraping and training models for this assignment.

References

[1] Kimberly Aguirre. 2024. *Letterboxd started as a cinephile's best-kept secret. Now studios want in.* Retrieved November 29, 2024 from https://www.latimes.com/entertainment-arts/story/2024-06-17/letterboxd-rise-connecting-filmmakers-studios-audiences

[2] Yu chen Fan, Yitong Ji, Jie Zhang, and Aixin Sun. 2024. Our Model Achieves Excellent Performance on MovieLens: What Does it Mean? arXiv:2307.09985 [cs.IR] https://arxiv.org/abs/2307.09985

[3] Hao Fan, Mengyi Zhu, Yanrong Hu, Hailin Feng, Zhijie He, Hongjiu Liu, and Qingyang Liu. 2024. TiM4Rec: An Efficient Sequential Recommendation Model Based on Time-Aware Structured State Space Duality Model. arXiv:2409.16182 [cs.IR] https://arxiv.org/abs/2409.16182

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017). arXiv:1703.04247 http://arxiv.org/abs/1703.04247

[5] Letterboxd.com. 2024. *Browse films.* Retrieved December 3, 2024 from https://letterboxd.com/films/popular/this/week/

[6] Letterboxd.com. 2024. *Film data.* Retrieved November 22, 2024 from https://letterboxd.com/about/film-data/

[7] Letterboxd.com. 2024. *Letterboxd brand.* Retrieved November 22, 2024 from https://letterboxd.com/about/brand/

[8] Letterboxd.com. 2024. *Popular reviewers.* Retrieved December 4, 2024 from https://letterboxd.com/reviewers/popular/

[9] Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. 2024. Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models. arXiv:2403.03900 [cs.IR] https://arxiv.org/abs/2403.03900

[10] Julian McAuley. 2022. *Personalized Machine Learning.* Cambridge University Press.

[11] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. 995–1000. https://doi.org/10.1109/ICDM.2010.127

[12] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine* 42, 3 (Nov. 2021), 7–18. https://doi.org/10.1609/aimag.v42i3.18140

A Appendix

Table 2. Dataset overview

Column	Type	Definition
user	str	Unique username, not displayed name, in the sense that "letterboxd.com/user" will predictably link to their page
movie	str	Displayed movie title
movie_year	int	Year the movie released, to better relate it to other datasets
rating	float	Rating for this viewing. 0: denotes the film was not rated by the user; else, 0.5-5.0 scale with 0.5 size steps
liked	int	Whether the user marked the film as "liked" for this viewing. 0: not liked; 1: liked
review_date	str	Date watched in the format YYYY-MM-DD
rewatched	int	Whether or not the viewing is a rewatch. 0: not a rewatch; 1: rewatch.
review	str	Review text
num_likes	int	Number of likes received by the review
tags	list(str)	List of tags provided for the review
liked_reviews	list(str)	List of links to reviews for this film the user liked
review_link	str	Link to movie review
movie_name	str	Display name
movie_id	str	ID used to reference it on Letterboxd in the format "letterboxd.com/film/movie_id/"
movie_year	int	Year released
movie_tagline	str	Tagline/subtitle
movie_desc	str	Description
movie_len	int	Length in minutes
genres	list(str)	List of genres in decreasing significance
top_keywords	list(str)	Top listed keywords in decreasing significance.
in_collection	int	If the film is part of a collection (or series). 0: not in a collection; 1: in a collection
collection_name	str	If `in_collection` is True, the name of the collection
studios	dict(str: str)	List of studio identifiers to display name
countries	dict(str: str)	Mapping of country identifiers to display name
primary_language	str	Primary language of the movie
spoken_languages	list(str)	List of languages spoken throughout the movie
imdb_id	int	Unique IMDb identifier. 0: no unique ID
tmdb_id	int	Unique TMDb identifier. 0: no unique ID
cast	dict(str: list(str))	Mapping of cast members to a list of their held positions.
crew	dict(str: str)	Mapping of cast members to their character name
num_watched	int	Number of users who watched the film
num_listed	int	Number of users who added the film to a list
num_liked	int	Number of users who 'liked' the film
top_250_rank	int	Position in Letterboxd's Top 250 Narrative Feature Films list, if applicable. 0: not in the Top 250 list
num_half_star	int	Number of total ½ star ratings
num_one_star	int	Number of total 1 star ratings
num_one_half_star	int	Number of total 1 ½ star ratings
num_two_star	int	Number of total 2 star ratings
num_two_half_star	int	Number of total 2 ½ star ratings
num_three_star	int	Number of total 3 star ratings
num_three_half_star	int	Number of total 3 ½ star ratings
num_four_star	int	Number of total 4 star ratings
num_four_half_star	int	Number of total 4 ½ star ratings
num_five_star	int	Number of total 5 star ratings