Explanations for Temporal Recommendations

Homanga Bharadhwaj1 · Shruti Joshi2

Received: 10 September 2018 / Accepted: 17 September 2018 / Published online: 27 September 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract
Recommendation systems (RS) are an integral part of artificial intelligence (AI) and have become increasingly important in the growing age of commercialization in AI. Deep learning (DL) techniques for RS provide powerful latent-feature models for effective recommendation but suffer from the major drawback of being non-interpretable. In this paper we describe a framework for explainable temporal recommendations in a DL model. We consider an LSTM based Recurrent Neural Network architecture for recommendation and a neighbourhood based scheme for generating explanations in the model. We demonstrate the effectiveness of our approach through experiments on the Netflix dataset by jointly optimizing for both prediction accuracy and explainability.

Keywords Recommendation systems · Explainable AI · Recurrent Neural Networks

1 Introduction

Explainability in machine learning models [1, 2] has been a topic of intense research and debate. The issue of explainability in recommendation systems (RS) [3–7] is all the more pertinent due to their sheer ubiquity. RS have become embedded in all forms of user-interface interaction and now form the core of our everyday activities. So, to improve users’ experience and trust, transparency and explainability are increasingly being incorporated in practical recommender systems. For instance, Netflix justifies the movies recommended by displaying similar movies obtained through the social network of users [8]. Similarly, Amazon justifies its recommendations by showing similar items obtained through neighbourhood based Collaborative Filtering (CF) [9]. There has been growing research in Deep Learning (DL) based models for recommendations which are known for giving excellent prediction accuracies. Now there is almost a universal consensus regarding the fact that the main drawback of Deep Learning models is their non-interpretability, however there have been recent efforts towards mitigating this drawback such as [10, 11].

For RS, most explanation based methods either fall into the classical neighbourhood based Collaborative Filtering (CF) [12, 13] or rule-based methods [14, 15], or combine both as in [16]. CF based recommendation methods, which leverage the wisdom of the crowd are more popular due to their scalability and robustness. Some recent works such as [17, 18] have integrated explanations into Matrix Factorization (which is a latent factor model), and into Restricted Boltzmann Machines respectively. These however cannot be applied to temporal recommendations which seek to model user preferences over time [19–21]. Modeling temporal evolution of user preferences and item states for effective recommendation systems (RS) is an active area of research and recent publications have illustrated the effectiveness of Recurrent Neural Networks (RNN) [22, 23] for the same.

Recurrent Recommender Networks [22] is a powerful technique of temporal recommendations. We build our model on top of this architecture by incorporating a neighbourhood-style explanation scheme. Here, LSTM [24] based RNNs are used for modelling user and item latent states. The specific domain of movie recommendation is targeted in the experiments but the method is fairly generalizable across domains. We first formalize the notion of explainability by defining a time-varying bipartite graph between users and items such that the edge weights measure a notion
of explainability of an item for a user by exploiting ratings of other users similar to the one in question. To optimize for explainability in addition to the prediction accuracy, we include a term in the optimization objective that seeks to minimize the distance between latent features of items and users weighed by their explainability as defined previously.

2 Related Works

2.1 Explainable Recommendations

Explanations in recommendation systems has been a topic of active research for a very long time, motivated by the general consensus that modern RS algorithms have been black boxes offering no transparency or human-interpretable insights. Although the underlying algorithm of a recommendation framework may influence the type of explanations generated, it is also an ecologically valid method to have a completely separate engine for generating explanations [5]. This is particularly interesting for complex RS models like those using collaborative filtering and/or deep learning techniques [25, 26]. Some methods generate explanations based on auxiliary data like review texts [27] while others do not require additional information (apart from that used in the recommendation algorithm) for generating explanations [17, 18, 25].

2.2 Temporal Recommendations

The temporal evolution of user preferences and item states has been discussed in multiple previous works. Recent papers which have been very impactful include [28] and RRN [22]. Donkers [28] developed a Gated Recurrent Unit (GRU) [29] based RNN for sequential modelling of user preferences. A rectified linear integration of user states is done in their modified GRU cell to evolve each user state for personalized recommendation. On the other hand, RRN targeted movie recommendation tasks by exploiting explicit feedback from users. Customized LSTM cells were used for modelling the function through which user and item latent states evolve. [30] leverage sequence information to mitigate the scarcity of user-item interactions and identify items that will be eventually preferred and that those that will be immediately desired. Since our architecture is built over RRN, we elaborate on the details in Sect. 3.2.

3 The Model

3.1 Explanations

Our model uses only the users’ movie rating data for predicting as well as explaining recommendations. This is in contrast to many previous studies that consider other data variables such as user demographics, text reviews [27], user preferences for entities associated with the items to be recommended [31], etc. for predicting recommendations, as well as studies that use these data variables as auxiliary information for only explaining their recommendations [25]. We employ a neighbourhood based explanation scheme which is similar to [18] and formalize the definition of neighbourhood and explainability as follows.

Neighbourhood is calculated based on discounted cosine similarity between the users’ (say user $i$ and user $k$) as $\text{sim}_{i,k} = \sum r_{im}^t r_{km}^t$, where the discounting factor, $γ_t = \frac{1}{1+t}$ and $r_{im}^t$ represents user $i$’s rating at time $t$ for a movie $m$. We then pick the $p$ most similar users as the neighbourhood of user $i$.

This notion of a neighbourhood style explanation has also been employed in various graph based models such as [31]. We define a temporally varying bipartite graph between the set of users and the set of items with the following edge weight matrix

$$M_{umt} = \frac{\sum_{z\in Q'(u)} r_{zm}^t}{p \times \max \{ r_{zm}^t \forall z \in Q'(u) \}}. \tag{1}$$

Here $Q'(u)$ is the set of $p$ neighbours for user $u$ and $r_{zm}^t$ is the rating of user $z$ for movie $m$ at time step $t$. It is important to note that $M_{umt}$, which represents the edge-weight between user $u$ and movie $m$ at time step $t$ is a real number between 0 and 1. It has the interpretation of being a quantification of how explainable is movie $m$ for user $u$. $r_{zm}^t$ is 0 if user $z$ has not rated item $m$ at time step $t$. So, a value of $M_{umt} = 0$ would mean that none of the users in the neighborhood of user $u$ have rated movie $m$ at time step $t$ and hence movie $m$ is not explainable to user $u$ at that time step.

In addition, since we postulate that there are stationary components as well in the latent features of users and items (Sect. 3.2) we also develop a stationary bipartite graph between the set of users and the set of items for explanations. This is in addition to the above time varying interpretation. For this bipartite graph, the edge weight matrix is described as

$$M_{um} = \frac{\sum_{z\in Q'(u)} r_{zm}}{p \times \max \{ r_{zm} \forall z \in Q'(u) \}}. \tag{2}$$

Here, all the terms have the same meaning as in the previous equation, but they appear without the time index.

Figure 2 shows some of the explanations generated by our model. The explanations are in the form of time-varying neighborhood of users, that show how the preference of other users similar to the user being considered vary over time. The twin effects of recency of ratings by similar users
and the sum of ratings have a tradeoff in the explanations as demonstrated in Fig. 2.

### 3.2 The Rating Prediction Model

We use LSTM based Recurrent Neural Networks for modelling the temporal evolution of user and movie latent states. The approach is essentially matrix factorization through time with the temporal evolution of latent factors being modelled by LSTM based Recurrent Neural Networks. This approach is similar in spirit to RRN [22] and the novelty of our method is the incorporation of explainability which we describe in Sect. 3.3.

We denote by \( u_i \) and \( m_j \), the latent features of user \( i \) and movie \( j \) respectively at time index \( t \). Let \( \hat{r}_{ij,t} \) denote the predicted rating for user \( i \) on movie \( j \) at time index \( t \) and let \( r_{ij,t} \) be the actual rating. \( \{r_{ij,t}\}_j \) denotes the set of ratings for all movies \( j \). We define the following model for updates

\[
\begin{align*}
\hat{r}_{ij,t} &= f(u_{it}, m_{jt}) \\
u_{i,t+1} &= g(u_{it}, \{r_{ij,t}\}_j) \\
m_{j,t+1} &= h(m_{jt}, \{r_{ij,t}\}_j).
\end{align*}
\]

The functions \( f, g, h \) are learned by the model to infer user and movie states. Section 3.1 of the RRN paper [22] elaborates on the user and movie state formulation, which we mention briefly below.

\( y_t = V[x_t, 1, \tau_t, \tau_{t-1}], u_t = LSTM(u_{t-1}, y_t). \)

Here \( \tau \) represents the wall clock at time \( t \), \( x_t \) is the rating vector for the user and \( V \) represents the transformation. Similar to the approach followed in RRN [22], we include the profile identities of the user and the movie as the stationary components of the rating in addition to their time varying state vectors. So,

\[
\begin{align*}
\hat{r}_{ij,t} &= f(u_{it}, m_{jt}, u_i, m_j) \\
&= (\tilde{u}_i, \tilde{m}_j) + (u_i, m_j)
\end{align*}
\]

where \( \tilde{u}_i = Au_i + c \) and \( \tilde{m}_j = Bm_j + d \). This decomposition makes it evident how RRN is essentially matrix factorization through time as mentioned earlier.

### 3.3 Incorporating Explainability in the Model

Since the user and item (movie) states are time varying, we need a time varying bipartite graph which is defined by a time varying edge-weight matrix \( M_{ij} \). If movie \( j \) is explainable to user \( i \) at time step \( t \), then their latent representations \( \tilde{m}_j \) and \( \tilde{u}_i \) respectively must be close. Based on this intuition, we include the term \( (\tilde{m}_j - \tilde{u}_i)^2 M_{ij} \) in our minimization objective. We formulate the overall objective in such a way that both prediction accuracy as well as explainability are optimized. So, if there are two movies which are likely to be equally preferred by the user, the model will recommend the movie which is more explainable. It is important to note that explainability and prediction accuracy may be at odds for some user-movie pairs and hence we need to define a joint objective function including both the aspects, which is defined as follows

\[
L = \sum_{ij} \left( \sum_t (r_{ij,t} - \hat{r}_{ij,t}(\theta))^2 + \alpha(\hat{m}_{ij} - \hat{u}_{ij})^2 M_{ij} \right) + \beta(m_{ij} - u_{ij})^2 M_{ij} + R(\theta).
\]

The benefit of having temporally varying explanation-graphs is that the generated explanations aren’t the conventional “7/10 people with similar interests as you have rated this movie 4 and higher” but can employ information related to rating distribution across time too, as seen in Fig. 1.

If we use the heuristic that explanations in the near past are more relevant than those in the far past, we can weigh the term \((\hat{m}_{ij} - \hat{u}_{ij})^2 M_{ij}\) for explanations by a temporally decaying factor \( \alpha(t) \). In this paper, we use the specific form of \( \alpha(t) \) to be \( \exp(-\alpha t) \) but there are other popular choices of this discount factor as well [32]. So, the modified objective function becomes

\[
L’ = \sum_{ij} \left( \sum_t (r_{ij,t} - \hat{r}_{ij,t}(\theta))^2 + \exp(-\alpha t)(\hat{m}_{ij} - \hat{u}_{ij})^2 M_{ij} \right) + \beta(m_{ij} - u_{ij})^2 M_{ij} + R(\theta).
\]

Here, we call the model corresponding to the objective function \( \text{L} \) as TempEx-Dry and the one corresponding to \( \text{L’} \) as TempEx-Fluid. We then perform simulations for both the objective functions and compare their pros and cons empirically.

### 3.4 Training

Although the conventional method of training Recurrent Neural Networks is Backpropagation Through Time (BPTT), as mentioned in [22], backpropagation through two sequences (rating depends on both user state RNN and item state RNN) is computationally infeasible. We adopt the strategy of subspace descent as done in [22] to alleviate this problem. In practice, we found that using Dropout [33] helps in stabilizing the gradients and preventing over-fitting due to the additional terms introduced in the minimization objective. The hyperparameters \( \alpha \) and \( \beta \) were tuned by a grid-search in the range 0 to 1 and the tuned value of \( \alpha \) is kept at 0.4, while that of \( \beta \) is kept at 0.6 throughout all experiments.
Fig. 1 Shown above are two instances of top-3 recommendations, decreasing left to right in confidence. Here, Epoch 1: within a month before present, Epoch 2: within 1 year before present, and Epoch 3: ≥ 1 year before present. In the top row, dominance of recency in ranking is see, while in the bottom row, the dominance of the sum of the ranking is evident. These are anecdotal examples from the evaluation of TempEx-Fluid on the Netflix dataset.

Fig. 2 MEP, MER, mean average precision (MAP) and MR (mean recall) for benchmark models with varying number of neighbours. All the values are averaged over the test set for all users.
4 Simulation Studies

4.1 Setup

Through a series of simulation experiments, we seek to understand two basic questions, (1) How effective is the model in generating explanations? and (2) What is the trade-off between prediction accuracy and explainability? All our experiments have been done using Tensorflow r1.4 [34] in Python 3. We use ADAM optimizer during training [35]. We perform all experiments on a timestamped Netflix dataset used in [22], which was first used in [8]. It consists of 100M movie ratings from 1999 to 2005, where each data point is a (user-id, item-id, time-stamp, rating) tuple with a time stamp granularity of 1 day. For consistency, we use the same preprocessing and train-test split as in [22].

4.2 Benefit of Explanations

To answer (1), we use standard IR metrics like precision and recall with the notion of explainability. [18] introduces Mean Explainable Precision (MEP) and Mean Explainable Recall (MER) metrics. To state briefly, MEP is defined as the ratio of the number of explainable items recommended to the total recommended items for each user averaged over all users. Similarly, MER is the number of explainable items recommended to the total number of explainable items for each user averaged over all users. We benchmark our performance against state of the art models such as RRN, T-SVD, PMF and recent explainable CF methods like EMF and ERBM [17, 18, 22, 36, 37]. We also evaluate two versions of our model—without incorporation of the temporally weighted explanation term (TempEx-Dry), and with an exponentially decaying temporal weight on explanations (TempEx-Fluid).

As revealed in the top row of Fig. 2, TempEx-Fluid has the best measure of explainability across all values of $p$ and it also performs consistently better than TempEx-Dry. So, temporally weighing the explainability term indeed leads to a more explainable model.

4.3 Tradeoff Between Explanations and Prediction Accuracy

To answer (2), we evaluate the performance of our model against benchmark models on standard metrics like RMSE, mean average precision (MAP), mean recall (MR) and mean reciprocal rank (MRR). The purpose of this evaluation is to see whether incorporation of the explainability terms in the optimization objective leads to substantial gains/losses on the actual prediction accuracy of ratings.

Table 1 and the bottom row of Figure 2 shows the results of our analysis. At $p = 50$ we observe that the RMSE and MRR values of TempEx-Fluid are higher than the standard RRN model. This indicates that incorporating explainability also improves the prediction accuracy on the test set by imposing an additional regularizer in the model. The bottom row of Fig. 2 shows that TempEx-Fluid consistently has better performance on Mean Average Precision and Mean Recall for all values of $p$. This leads further credence to the fact that there is no compromise being made on prediction accuracy by including explainability. Also, a comparison between TempEx-Dry and TempEx-Fluid reveals that TempEx-Fluid performs consistently better for all values of $p$. This is due to the more nuanced temporal decay in the optimization objective which appropriately weighs down past effects.

5 Conclusion

In this paper, we devised a methodology of incorporating explanations in time-series recommendation. We devised a time-varying neighbourhood style explanation scheme and jointly optimized for prediction accuracy and explainability. Through simulation results we demonstrated the efficacy of the proposed framework. It is important to note that our method of explanation is different from the core recommendation algorithm, which is a common practice in explainable RS. However, as future work we plan to devise an explanation scheme that tries to explain the recommendation algorithm. Since, our algorithm is a deep learning model, we need to incorporate schemes like layer-wise relevance propagation that seek to propagate the relevance of the output through the layers of the network and assign relevance to the inputs.

References

1. Turner R (2016) A model explanation system. In: 2016 IEEE 26th international workshop on machine learning for signal processing (MLSP). IEEE, New York, pp 1–6
2. Samek W, Wiegand T, Müller KR (2017) Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. arXiv preprint. arXiv:1708.08296
3. Resnick P, Varian HR (1997) Recommender systems. Commun ACM 40(3):56
4. Ricci F, Rokach L, Shapira B (eds) (2015) Recommender systems: introduction and challenges. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, Berlin, pp 1–34
5. Tintarev N, Masthoff J (2011) Designing and evaluating explanations for recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender systems handbook. Springer, Berlin, pp 479–510
6. McSherry D (2005) Explanation in recommender systems. Artif Intell Rev 24(2):179
7. Tintarev N, Masthoff J (2007) A survey of explanations in recommender systems. In: 2007 IEEE 23rd international conference on data engineering workshop. IEEE, New York, pp 801–810
8. Diao Q, Qiu M, Wu CY, Smola AJ, Jiang J, Wang C (2014) Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 193–202
9. Smith B, Linden G (2017) Two decades of recommender systems at Amazon.com. IEEE Internet Comput 21(3):12
10. Binder A, Bach S, Montavon G, Müller KR, Samek W (2016) Layer-wise relevance propagation for deep neural network architectures. In: Kim K, Joukov N (eds) Information science and applications (ICISA) 2016. Springer, Berlin, pp 913–922
11. Zhang Y, Wallace B (2015) A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. In: Proceedings of the eighth international joint conference on natural language. Asian Federation of Natural Language Processing, Taipei, pp 253–263
12. Schafer JB, Frankowski D, Herlocker J, Sen S (2007) Collaborative filtering recommender systems. In: Brusilovsky P, Kobza A, Nejdl W (eds) The adaptive web. Springer, Berlin, pp 291–324
13. Ekstrand MD, Riedl JT, Konstan JA et al (2011) Collaborative filtering recommender systems. Found Trends Hum Comput Interact 4(2):81
14. Pazzani MJ, Billsus D (2007) Content-based recommendation systems. In: Brusilovsky P, Kobza A, Nejdl W (eds) The adaptive web. Springer, Berlin, pp 325–341
15. Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender systems handbook. Springer, Berlin, pp 73–105
16. Burke R (2002) Hybrid recommender systems: survey and experiments. User Model User Adapt Interact 12(4):331
17. Abdollahi B, Nasraoui O (2016) In: Explainable matrix factorization for collaborative filtering. In: Proceedings of the 25th international conference companion on world wide web (international world wide web conferences steering committee, 2016), pp 5–6, April 11–15 2016
18. Abdollahi B, Nasraoui O (2016) Explainable restricted Boltzmann machines for collaborative filtering. arXiv preprint. arXiv :1606.07129
19. Liu K, Shi X, Kumar A, Zhu L, Natarajan P (2016) Temporal learning and sequence modeling for a job recommender system. In: Proceedings of the recommender systems challenge. ACM, New York, p 7
20. Lathia N, Hailles S, Capra L, Amatriain X (2010) Temporal diversity in recommender systems. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 210–217
21. Ullah F, Sarwar G, Lee SC, Park YK, Moon KD, Kim JT (2012) Hybrid recommender system with temporal information. In: 2012 International conference on information networking (ICOIN). IEEE, New York, pp 421–425
22. Wu CY, Ahmed A, Beutel A, Smola AJ, Jing H (2017) Recurrent recommender networks. In: Proceedings of the tenth ACM international conference on web search and data mining. ACM, New York, pp 495–503
23. Hidasi B, Quadrama M, Karatzoglou A, Tikk D (2016) Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: Proceedings of the 10th ACM conference on recommender systems. ACM, New York, pp 241–248
24. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735
25. Herlocker JL, Konstan JA, Riedl J (2000) Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM conference on computer supported cooperative work. ACM, New York, pp 241–250
26. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Eighth IEEE international conference on data mining, 2008. ICDM’08. IEEE, New York, pp 263–272
27. Zhang Y (2015) Collaborative filtering for recommender systems. In: Proceedings of the eighth ACM international conference on web search and data mining. ACM, New York, pp 435–440
28. Donkers T, Loepp B, Ziegler J (2017) Sequential User-based recurrent neural network recommendations. In: Proceedings of the eleventh ACM conference on recommender systems. ACM, New York, pp 152–160
29. Cho K, Van Merriënoor B, Bahdaanu D, Bengio Y (2014) On the properties of neural machine translation: encoder-decoder approaches. In: Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation. Association for Computational Linguistics, SSST
30. Devoodt R, Bersini H (2017) Long and short-term recommendations with recurrent neural networks. In: Proceedings of the 25th conference on user modeling, adaptation and Personalization. ACM, New York, pp 13–21
31. He X, Chen T, Kan MY, Chen X (2015) TriRank: review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM international conference on information and knowledge management. ACM, New York, pp 1661–1670
32. Krogh A, Hertz JA (1992) A simple weight decay can improve generalization. In: Advances in neural information processing systems. Morgan Kaufmann, pp 950–957
33. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929
34. Abadi M, Barham P, Chen J, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) TensorFlow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX conference on operating systems design and implementation. USENIX Association Berkeley, Savannah, pp 265–283
35. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint. arXiv :1412.6980
36. Mnih A, Salakhutdinov RR (2008) Probabilistic matrix factorization. In: Proceedings of the 20th International Conference on Neural Information Processing Systems. Curran Associates Inc., British Columbia, pp 1257–1264
37. Koren Y (2010) Collaborative filtering with temporal dynamics. Commun ACM 53(4):89