JOINT TRAINING OF RATINGS AND REVIEWS WITH RECURRENT RECOMMENDER NETWORKS

Chao-Yuan Wu
University of Texas at Austin
Austin, TX, USA
cywu@cs.utexas.edu

Amr Ahmed & Alex Beutel[*]
Google
Mountain View, CA, USA
{amra,alexbeutel}@google.com

Alexander J. Smola
Carnegie Mellon University
Pittsburgh, PA, USA
alex@smola.org

ABSTRACT

Accurate modeling of ratings and text reviews is at the core of successful recommender systems. While neural networks have been remarkably successful in modeling images and natural language, they have been largely unexplored in recommender system research. In this paper, we provide a neural network model that combines ratings, reviews, and temporal patterns to learn highly accurate recommendations. We co-train for prediction on both numerical ratings and natural language reviews, as well as using a recurrent architecture to capture the dynamic components of users' and items' states. We demonstrate that incorporating text reviews and temporal dynamic gives state-of-the-art results over the IMDb dataset.

1 INTRODUCTION

Designing highly accurate recommender systems has been the focus of research in many communities and at the center of many products for the past decade. The core goal is to predict which items a given user will like or dislike, typically based on a database of previous ratings and reviews. In particular, a good recommender system has been defined as one that predicts the rating for randomly chosen and unseen (*user*,*item*) pairs. During the Netflix Prize contest, a variety of factorization models were proposed to capture the latent embeddings of users and items that would lead to accurate recommendations (Bell & Koren, 2007; Koren et al., 2009). Generative models for personalized ratings have recently become popular, due to impressive and robust results (Mnih & Salakhutdinov, 2007; Salakhutdinov & Mnih, 2008; Stern et al., 2009; Beutel et al., 2015).

More recently, there has been an interest in the recommender system community to also make use of the rich natural language reviews provided by users. Most often, these reviews have been transformed into a bag-of-words-model and used as a sort of regularization for the rating predictions (McAuley & Leskovec, 2013; Diao et al., 2014; Almahairi et al., 2015; Wu et al., 2016b). Using reviews in this way has been found to improve prediction accuracy, and in some cases provide detailed explanations for the recommendations.

This previous research has been remarkably successful, but has two significant limitations that we discuss and address in this paper. First, prediction accuracy has rarely been measured by the ability of a model to predict *future* ratings. Rather, recommendation accuracy has been derived from a random split of the ratings data, which undermines our understanding of the models' usefulness in practice. Here, we focus on predicting future ratings, splitting our training and testing data by date. In order to be successful at this task, we incorporate the time of ratings and reviews in our model structure and training. Koren (2010) previously derived temporal features of ratings data, but used these features to *remove* temporal effects since the metric of success was interpolation, not extrapolation. More recently, Recurrent Recommender Networks (RNN) use a recurrent neural network to capture changes

[*]A majority of this work was done while the author was at Carnegie Mellon University.

in both user preferences and item perceptions, and *extrapolate* future ratings in an autoregressive way (Wu et al., 2016a). However, temporal patterns in reviews are largely unexplored. Note that just like ratings, reviews also depend on *changing* factors, such as user writing styles, user preferences, movie perceptions, or the popularity of certain slang words or emoticons. Here we use a generative LSTM model that is able to jointly model the temporal effects in ratings and reviews.

Second, models of reviews in recommender system fall significantly behind the state-of-the-art in natural language processing. The bag-of-words model used in previous research improves over not using text, but is limited in the degree to which it can understand the review. In fact, the drawback of an underfitting model is especially salient in the case of reviews, because they are much more diverse and unstructured than regular documents. Recently there has been significant research attention on modeling natural language with neural networks, with encouraging results (Lipton et al., 2015; Yang et al., 2016). Here, we combine these powerful neural-based language models with recurrent neural network to learn both accurate recommendations and accurate reviews. Our main contributions are as follows:

- **Joint generative model:** We propose a novel joint model of ratings and reviews via interacting recurrent networks (particularly LSTM).
- **Nonlinear nonparametric review model:** By learning a function of user and movie state dynamics, we can capture the evolution of reviews (as well as ratings) over time.
- **Experiments** show that by jointly modeling ratings and reviews along with temporal patterns, our model achieves state-of-the-art results on IMDb dataset in terms of forward prediction, i.e. in the realistic scenario where we use only ratings strictly prior to prediction time to predict future ratings.

2 Related Work

Collaborative Filtering As mentioned in the introduction, recommender systems have been the focus of many different research communities. The Netflix Prize generated a flurry of research to improve recommendation accuracy, with a variety of matrix factorization models being proposed (Bell & Koren, 2007; Koren et al., 2009; Koren, 2008). During the Netflix competition and more afterwards, a stream of research has focused on designing generative Bayesian models for user ratings data (Mnih & Salakhutdinov, 2007; Salakhutdinov & Mnih, 2008; Stern et al., 2009; Beutel et al., 2014; 2015). Nearly all of these models predict ratings by an inner product between a latent user embedding and a latent item embedding; different approaches primarily regularization, e.g., Bayesian models and learning algorithms capture uncertainty in the data.

Other models have tried to capture interesting patterns discovered in ratings data. As an example, Beutel et al. (2014) finds that some ratings form bimodal rather than Gaussian distributions and designs a model to accommodate this diversity. More closely related to this work, Koren (2010) designs *many* features to capture and *remove* the temporal effects in ratings data. By removing these temporal effects, Koren (2010) learns better stationary embeddings for users and items. Work such as this improves prediction accuracy, but has two drawbacks: (1) it requires time consuming feature engineering, and (2) it focuses on interpolation rather than extrapolation into the future. Wu et al. (2016a) addresses both of these concerns by learning a function for the evolution of user preferences and item properties. However, this work focuses exclusively on modeling ratings over time and, in a large part, on the qualitative patterns discovered in the Netflix dataset. Here we focus on the model itself and, in particular, the interaction of jointly understanding ratings, reviews, and temporal patterns.

Review Modeling Although the most common metric for recommendation accuracy has been rating prediction, natural language reviews provide rich, detailed insight into user preferences. Most often, reviews have been used in a bag-of-words model to regularize rating prediction (McAuley & Leskovec, 2013; Diao et al., 2014; Wu et al., 2016b). For example, McAuley & Leskovec (2013) effectively learns a topic model of reviews regularize item embeddings. By using such coarse models, the impact of and insight from reviews is limited. More recently, Almahairi et al. (2015) use neural network based review models to regularize hidden factors, but their model assumes only stationary states.

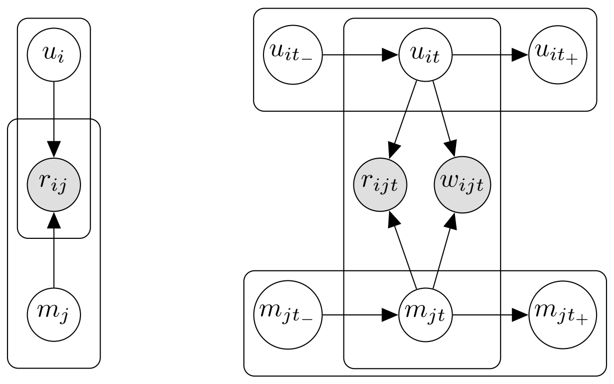


Figure 1: As shown on the left, previous recommendation models learn static stationary embeddings for users and movies to predict ratings. As shown on the right, we can also capture temporal effects present in the data. We have both user and movie embeddings follow a Markov chain, and use these dynamic embeddings (along with stationary ones not shown) to predict both ratings and text reviews.

Interestingly, data mining research has found that review patterns are dynamic, with different language being adopted by communities over time (Danescu-Niculescu-Mizil et al., 2013). Therefore, it is important to capture not just the dynamics of ratings, but also the language used to justify those ratings.

Neural Networks Neural networks have recently offered large improvements in natural language processing. More recently, a few papers have focused these natural language models on online reviews (Lipton et al., 2015; Yang et al., 2016). However, while these papers do model online reviews, they differ greatly from our work in that they are not actually used for recommendation.

With the recent remarkable successes of neural networks in other domains, there has been growing attention on using neural networks for model graphs and ratings data. Most similar, Sedhain et al. (2015) design an autoencoder for collaborative filtering.

LSTM and Recurrent Network Recurrent neural network provides a powerful tool to nonparametrically model temporal data by using a latent variable autoregressive model as follows:

$$\hat{z}_{t+1} = f(h_t, z_t) \text{ and } h_{t+1} = g(h_t, z_{t+1}).$$

Where z_t is the observation at time t, $\hat{z}_t$ is the model associated estimate, and h_t denotes the *latent state*. A popular class of RNN is the Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and we use this as a building block in our model .The state updates is given below:

$$[f_t, i_t, o_t] = \sigma \left[W \left[h_{t-1}, z_t\right] + b\right] \tag{1}$$
$$l_t = \tanh \left[V \left[h_{t-1}, z_t\right] + d\right] \tag{2}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{3}$$
$$h_t = o_t \cdot \tanh(c_t), \tag{4}$$

where f_t, i_t, o_t denote the forget gate, input gate and the output gate respectively. For simplicity in the following we denote this set of operations by $h_t = \text{LSTM}(h_{t-1}, z_t)$. We will refer to h_t as the output *embedding* from the LSTM.

3 Model

A comparison of our model with traditional recommender systems is illustrated in Figure 1. In previous recommender systems, ratings are assumed to be a function of *stationary* user and movie embeddings. Here we consider *dynamic* embeddings that predict both ratings and text reviews at a given time step.

Figure 2 shows a depiction of our model: Joint Review-Rating Recurrent Recommender Network. In addition to stationary embeddings as used in traditional recommender systems, here we use two

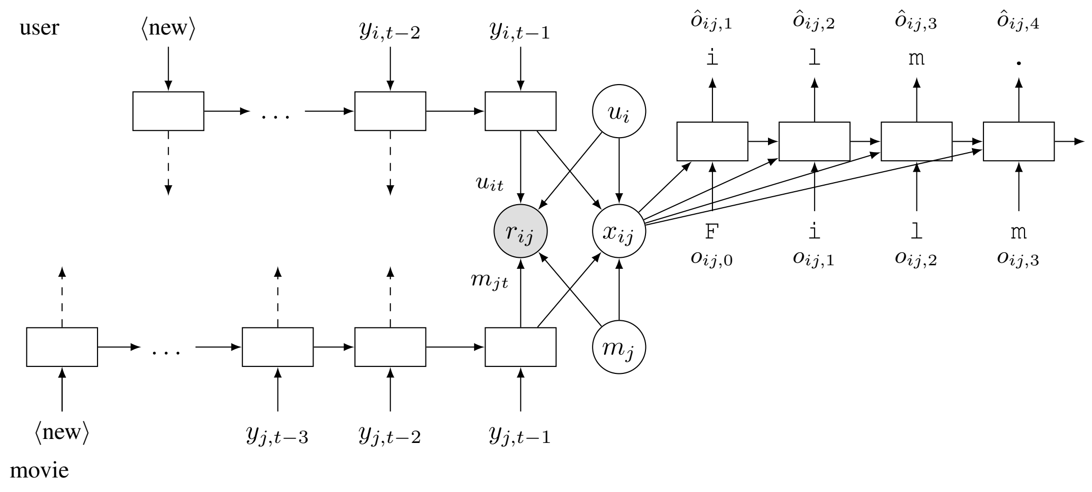


Figure 2: Joint Review-Rating Recurrent Recommender Networks: We use recurrent networks to capture the temporal evolution of user and movies states. The recurrent networks depend on the ratings of a user (and movie) in previous time steps. We combine these dynamic states with classic stationary states. We directly use all of these states to predict ratings, and use them within an LSTM to model review text.

LSTM RNNs that take user/movie history as input to capture the temporal dynamics in both user and movie states. Given stationary and dynamic states of user i and movie j, we define generator functions that emit both rating $r_{ij}|t$ and reviews $o_{ij}|t$ at time step t. Formally,

$$\begin{aligned} r_{ij}|t = f(u_i, m_j, u_{it}, m_{jt}) \quad &\text{and} \quad o_{ij}|t = \psi(u_i, m_j, u_{it}, m_{jt}) \\ u_{i,t+1} = g(u_{it}, \{r_{ij}|t\}) \quad &\text{and} \quad m_{j,t+1} = h(m_{jt}, \{r_{ij}|t\}), \end{aligned}$$

where u_i and m_j denote stationary states, and u_{it} and m_{it} denote the dynamic state at t. Note that with learned f, ψ, g and h and given user/movie history, an user/movie state can be *inferred* without further optimization. In other words, different from traditional recommender systems, here we learn the *functions* that find the states instead of learning the states directly.

3.1 Dynamic User and Movie State

Here we give a detailed description on the RNNs that find the dynamic states. The key idea is to use user/movie rating history as inputs to update the states. In this way we are able to model *causality* instead of just finding correlation. That is, we can model e.g. the change of user (movie) state caused by having watched and liked/disliked a movie (being liked/disliked by certain users). At each step, the network takes

$$y_t := W_{\text{embed}} \left[x_t, 1_{\text{newbie}}, \tau_t, \tau_{t-1} \right], \tag{5}$$

where x_t is the rating vector, 1_{newbie} is the indicator for new users, and τ_t is wall-clock time. The jth element of x_t is the rating the user gives for movie j at time t, and 0 otherwise. 1_{newbie} effectively select a default embedding for a new user, and τ_t and τ_{t-1} gives the model the information to synchronize between RNNs and model the effects such as rating scale change or movie age. Note that with the inclusion of τs, we do not need to include the steps where a user did not rate any movie, and this can drastically speed up training. The state update is given by standard $u_t := \text{LSTM}(u_{t-1}, y_t)$. In the above we omit user index for clarity. In cases where we need to distinguish different users (and movies) such as in Figure 2, we use additional index i for user i as in u_{it}, and similarly for movie j in m_{jt}.

3.2 Rating Emissions

We supplement the time-varying profile vectors u_{it} and m_{jt} with stationary ones u_i and m_j respectively. These *stationary* components encode time-invariant properties such as long-term preference of a user or the genre of a movie.

The review rating is thus modeled as a function of both dynamic and stationary states, i.e.

$$r_{ij} = f(u_{it}, m_{jt}, u_i, m_j) := \langle \tilde{u}_{it}, \tilde{m}_{jt} \rangle + \langle u_i, m_j \rangle \tag{6}$$

where $\tilde{u}_{it}$ and $\tilde{m}_{jt}$ are affine functions of u_{it} and m_{jt} respectively. That is, we have

$$\tilde{u}_{it} = W_{\text{user}} u_{it} + b_{\text{user}} \text{ and } \tilde{m}_{jt} = W_{\text{movie}} m_{jt} + b_{\text{movie}}$$

This makes the model a strict superset of popular matrix factorization recommender systems that accounts for stationary effects, while we use LSTMs, on top of that, to model longer-range dynamic updates.

3.3 Review Text Model

Review text is modeled by a character-level LSTM network. This network shares the same user/movie latent states with the rating model. After all, the purpose of a review is to explain its rating score. We fuse the stationary and dynamic states of both user of movie by the bottleneck layer $x_{\text{joint},ij}$ given below:

$$x_{\text{joint},ij} := \phi(W_{\text{joint}} \left[u_{it}, m_{jt}, u_i, m_j\right] + b_{\text{joint}}) \tag{7}$$

$$\tilde{x}_{ij,k} := \left[x_{o_{ij,k}}, x_{\text{joint},ij}\right] \tag{8}$$

where $o_{ij,k}$ denotes the character at position k for the review given by user i to movie j, and $x_{o_{ij,k}}$ denotes the embedding of the character. ϕ here is some non-linear function.

The review text emission model is itself an RNN, specifically a character-level LSTM generative model. For character index $k = 1, 2, \ldots,$

$$h_{ij,k} := \text{LSTM}(h_{ij,k-1}, \tilde{x}_{ij,k}) \tag{9}$$

$$\hat{o}_{ij,k} := \text{softmax}\left(W_{\text{out}} h_{ij,k} + b_{\text{out}}\right) \tag{10}$$

Here a softmax layer at output of LSTM is used to predict the next character. Generating text conditioned on contents has been applied to various areas, such as machine translation (Sutskever et al., 2014), question answering (Gao et al., 2015), or image captioning (Vinyals et al., 2015). Probably the most similar approach is Lipton et al. (2015), but it conditions review generation on observed ratings instead of latent states.

3.4 Prediction

In prediction time, we make rating predictions based on predicted future states. That is, we take the latest ratings as input to update the states, and use the newly predicted states to predict ratings. This differs from traditional approaches where embeddings are estimated instead of inferred.

3.5 Training

Our goal is to predict both accurate ratings and accurate reviews, and thus we minimize

$$L := \sum_{(i,j) \in \mathcal{D}_{\text{train}}} \left[\left(\hat{r}_{ij}(\theta) - r_{ij}\right)^2 - \lambda \sum_{k=1}^{n_{ij}} \log\left(\Pr(o_{ij,k}|\theta)\right) \right], \tag{11}$$

where $\mathcal{D}_{\text{train}}$ is the training set of (i, j) pairs, θ denotes all model parameters, and n_{ij} is the number of characters in the review user i gives to movie j. The first term corresponds to the deviation of the prediction from the actual rating, and the second term is the likelihood of the text reviews. λ controls the weight between predicting accurate ratings and predicting accurate reviews. Our training follows the subspace descent strategy in Wu et al. (2016a). That is, while the review generative model is updated in every iteration, the user-state and movie-state RNNs are updated in an alternating way.

Data			# users	# items	# ratings (reviews)	# characters
IMDb	Train	Jul 98 - Dec 12	6,127	8,002	402.3k	690.6M
	Test	Jan 13 - Sep 13			11.0k	21.6M
Netflix 6 months	Train	Jun - Nov 11	311.3k	17.7k	13.7M	-
	Test	Dec 11			2.1M	-

Table 1: IMDb dataset comprises reviews and ratings collected from July 1998 to September 2013. Netflix 6 months data is a subset of original Netflix prize dataset that is split based on time.

The gradients are calculated with standard backpropagation. Furthermore, we pre-warm train the review LSTM over the review text excluding the auxiliary input from the user and movie states. It is undesirable if the review likelihood overwhelms the rating. We hence normalize review likelihood by the number of characters in a review so that it does not dominates the rating likelihood. This technique is common in NLP literature (Wang & McCallum, 2006).

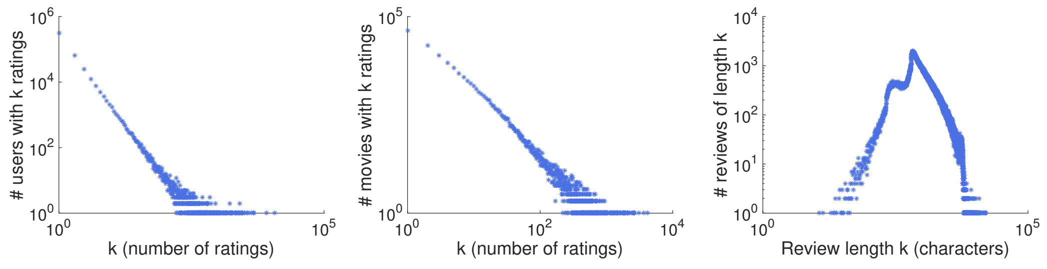
4 Experiments

In this section we empirically demonstrate the ability of our model to accurately predict both ratings and reviews, and capture temporal dynamics.

4.1 Experimental setup

In the following experiments, we select hyperparameters, optimization parameters and model architecture by cross-validation. The details are as follows. We use 1-layer LSTM recurrent neural networks with 40 hidden factors for user/movie state transitions. The input of this LSTM is an user/item embedding of dimension 40. Stationary and dynamic factors are 160 and 40-dimensional respectively. A 2-layer LSTM network is used to model texts, which takes 30-dimensional character embedding x_{char}, 40-dimensional state vector x_{joint}, and a 50-dimensional movie embedding x_{movie}.

To speed up convergence, we initialize the text model by a character-level RNN pre-trained without considering rating. Stationary factors are initialized by a pre-trained iAutoRec (Sedhain et al., 2015) model based on the last layer. We initialize all the other parameters from uniform distribution between $[-a, a]$ with $a = \sqrt{1.5(f_{in} + f_{out})}$, where f_{in} and f_{out} are fan-in and fan-out of transition matrices. ℓ_2 regularization with magnitude 0.001 is applied to all parameters. Dropout with a 0.5 rate is applied after all fully-connected layers. To prevent exploding gradients in of LSTM, gradients are clipped to $[-15, 15]$. ADAM (Kingma & Ba, 2014) with learning rate 0.0015 is used for optimization.



(a) User distribution. (b) Movie distribution. (c) Review length distribution.

Figure 3: Characteristics of IMDb dataset.

Data Here we focus on movie recommendations, where the opinions are highly dynamic. We evaluate our model on IMDb dataset, first used in Diao et al. (2014), that is the only large-scale movie

	PMF	Time-SVD++	U-AutoRec	I-AutoRec	RRN (rating)	RRN (rating + text)
IMDb	1.7355	1.7348	1.7332	1.7135	1.7047	**1.7012**
Netflix 6 months	0.9584	0.9589	0.9836	0.9778	**0.9427**	-

Table 2: RRN outperforms competing models in terms of RMSE. In addition, jointly modeling ratings and reviews achieves even better accuracy.

review dataset available. Restaurant recommendations (e.g. Yelp) could be also a suitable domain, but full rating history is not available in publicly available datasets[1].

The IMDb dataset contains full review and rating history of all users and all movies from 1998 to 2013. The characteristics of this dataset is shown in Figure 3. We see that the user and movie ratings follow heavy tail distributions, and thus the majority of users and movies have very few reviews, making accurate recommendation challenging for these users and movies. Review length is summarized in Figure 3 (c). Since one of the major goal of this project is to study temporal dynamics, we focus on users and items that have multiple interactions with the system. Specifically, we select a subset of k-core of the graph with $k = 15$. That is, each user and movie has at least 15 ratings in this subset. Note that the resulting subgraph is still very sparse – with only 0.8% density, which is sparser than for example, 1.2 % density of Netflix dataset . For completeness, we also include the 6-month Netflix dataset as used in Wu et al. (2016a), which has only ratings, to study RRN's ability to model temporal patterns.

The dataset is split by date instead of random sampling to simulate the real recommendation settings where we need to predict into the future instead of interpolating the past. IMDb training set contains all ratings from July 1998 to December 2012, and the ratings from January to September 2013 are randomly split into a validation set and a test set. Similarly, the 6-month Netflix dataset is split into January to November 2011 (training) and December 2011 (testing and validation). We report the results on testing set with the model that gives the best results on validation set. The summary of this dataset is given in Table 1.

Baselines We compare our model with models including the state-of-the-art temporal model, and a state-of-the-art neural network-based model.

- **PMF (Mnih & Salakhutdinov, 2007):** Our model extends matrix factorization by including a dynamic part and a joint review model. Comparing to PMF directly shows us the advantage of our approaches. LIBPMF (Yu et al., 2012) is used in experiments.
- **Time-SVD++ (Koren, 2010):** Time-SVD++ is the state-of-the-art model for temporal effects. It achieves excellent performance in Netflix contest. Implementation in GraphChi (Kyrola et al., 2012) is used in experiments.
- **AutoRec (Sedhain et al., 2015):** AutoRec is the state-of-the-art neural network recommender system. It learns an autoencoder that encodes user (item) histories into a low-dimensional space and then predict ratings by decoding. No temporal effects or causality are considered in this model. We use the software the authors provide in experiments.

All models use comparable number of factor sizes. Parameters of PMF and Time-SVD++ are selected by grid-search. Settings of AutoRec follow the original paper. We also include the performance of rating-only RRN, as in Wu et al. (2016a), to separate the benefits obtained from temporal modeling and review texts.

4.2 Rating prediction

One important goal of recommender systems is making accurate rating predictions. Here we evaluate the accuracy by root-mean-square error (RMSE) of prediction from the true rating. The results are summarized in Table 2. For completeness, we include the results from Wu et al. (2016a) on

[1] https://www.yelp.com/dataset_challenge

6-month Netflix dataset that use ratings only to compare the behavior of different models on different datasets. We see that rating-only RRN outperforms all baseline models in terms of rating prediction consistently in both dataset. More importantly, **joint-modeling ratings and reviews boosts the performance even more**, compared to rating-only RRN. This implies that by sharing statistical strength between ratings and reviews, the rich information in reviews helps us estimate the latent factors better. Note that while the absolute improvements in RMSE might not appear to be huge, the 1.98% improvement over PMF is actually considerable in terms of recommendations[2]. We also see that while Time-SVD++ performs well in Netflix contest, it does not work as well for predicting future ratings. After all, the goal of Time-SVD++ is estimating the temporal bias in hindsight instead of extrapolating into future states.

4.3 Text modeling

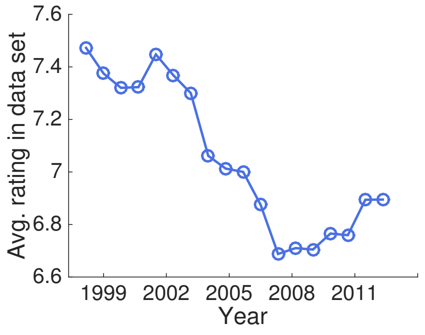
Here we examine the impact of conditioning on user and item states for text modeling. Towards this end, we compare perplexity of characters in testing set with and without using the user/item factors. Perplexity is defined as

$$\text{ppx}(D_{test}) = \exp\left(-\frac{1}{N_c}\sum_{c\in D_{test}} \log \Pr(c)\right),$$

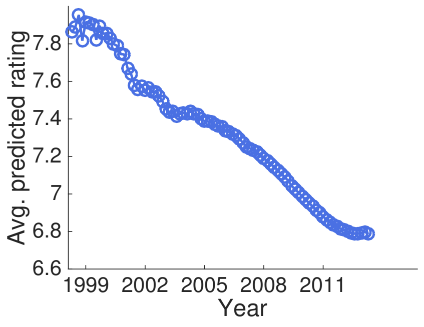
where N_c is the total number of characters in D_{test}, and $\Pr(c)$ is the likelihood of character c. Interestingly, we found that by jointly training with user and item states, the perplexity **improves** from 3.3442 to **3.3362**.

4.4 Temporal dynamics

Here we study if RRN is able to automatically capture the overall rating trends in IMDb by adaptively updating states along history sequence. Specifically, at each time step, we randomly sample up to 1000 users, and see what ratings the users would have given to each of the movie given their states at the time step, even in reality the user might not have given a rating to the movie. This gives us an unbiased estimation of average behavior of our model on each of the ratings. Figure 4 shows the average predicted ratings in this setting and the true average rating in the data set. We see that RRN clearly captures the overall trend in IMDb smoothly.



(a) Average ratings on IMDb.



(b) Predicted ratings.

Figure 4: RRN is able to capture the overall trend of data. (a) show the average ratings of all movies on IMDb over time. In (b) we see the predicted ratings are consistent with this trend.

5 Discussion & Conclusion

We present a novel approach that jointly models ratings, reviews, and their temporal dynamics with RRN. The contributions we have provided are as follows:

[2]For example, in 2009 SVD++ outperforms SVD by 1.09% and Time-SVD++ outperforms SVD++ by 1.25%, and they are considered important progress in recommender systems.

1. **Joint rating-review modeling:** We offer an LSTM-based joint rating-review model that provides advantages in both rating prediction and text modeling.
2. **Nonparametric dynamic review modeling:** RRN is based on an autoregressive method to model temporal dynamics of users and movies, allowing us to capture how reviews change over time.
3. **Empirical results:** We demonstrate that our joint model offers state-of-the-art results on rating prediction in real recommendation settings, i.e. predicting into the future.

REFERENCES

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 147–154. ACM, 2015.

R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations*, 2007. URL `http://doi.acm.org/10.1145/1345448.1345465`.

Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J Smola. Cobafi: collaborative bayesian filtering. In *WWW*, 2014.

Alex Beutel, Amr Ahmed, and Alexander J Smola. ACCAMS: Additive Co-Clustering to Approximate Matrices Succinctly. In *WWW*, 2015.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. No country for old members: User lifecycle and linguistic change in online communities. In *WWW*, 2013.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *KDD*. ACM, 2014.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. In *NIPS*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008. URL `http://doi.acm.org/10.1145/1401890.1401944`.

Y. Koren, R.M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 2009. URL `http://doi.ieeecomputersociety.org/10.1109/MC.2009.263`.

Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4): 89–97, 2010.

Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. In *OSDI*, 2012.

Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. Capturing meaning in product reviews with character-level generative text models. *CoRR*, 2015.

J. McAuley and J. Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *RecSys*, 2013. URL `http://doi.acm.org/10.1145/2507157.2507163`.

Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 2007.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In W.W. Cohen, A. McCallum, and S.T. Roweis (eds.), *ICML*, 2008.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *WWW Companion*, 2015.

David H Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*. ACM, 2009.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD*. ACM, 2006.

C.-Y. Wu, A. Beutel, A. Ahmed, A. J. Smola, and H. Jing. Recurrent recommender networks. In *Web Science and Data Mining (WSDM)*, 2016a.

Chao-Yuan Wu, Alex Beutel, Amr Ahmed, and Alexander J. Smola. Explaining reviews and ratings with PACO: poisson additive co-clustering. In *WWW Companion*, 2016b. doi: 10.1145/2872518.2889400. URL http://doi.acm.org/10.1145/2872518.2889400.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL*, 2016.

Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, 2012.