Hierarchical Temporal-Contextual Recommenders

Yifei Ma
Amazon AI
yifeim@amazon.com

Murali Balakrishnan Narayanaswamy
Amazon AI
muralibn@amazon.com

Abstract

Recommendation systems have developed beyond simple matrix factorization to focus on two important sources of information: the temporal order of events (Hidasi et al., 2015) and side (e.g., spatial) information encoded in user and item features (Rendle, 2012). However, state-of-art temporal modeling is often limited by model capacity for long user histories. In addition, meta data are rarely used in generic sequence models, perhaps due to a lack of improvement guarantees in end-to-end training. Important kinds of meta-data, like interaction feedback (e.g. click vs. add to cart, view duration) are not modeled. In this paper we propose a hierarchical recurrent network with meta data (HRNN-meta) model to solve both problems. To compactly store long histories, and propagate gradients through them, we use HRNN to group user interactions into hierarchical sessions of activity intervals, within which a user tends to maintain related interests. Different from previous hierarchical models (Quadrana et al., 2017), which manipulate model hidden states, HRNN encodes session information in the embedded inputs. We show that this change not only yields up to 10x better computational efficiency due to better ability to align batches, but also allows us to extend from GRUs (Cho et al., 2014) to the entire family of RNN models, and further increases model capacities when combined with temporal convolutional networks. To use meta data in sequential models, we extend the HRNN decoder with a factorization machine inspired network, between the HRNN output embedding and item meta data, which improves over the vanilla HRNN which is a special case of the model. We also extend HRNN-meta model to handle user features and interaction feedback to learn different objectives such as click or rating predictions. We report significant improvements both in simulation studies and in real-world datasets.

1 Introduction

Personalized recommendation systems typically use two sources of information: collaborative filtering based on user-item interaction histories (Shi et al., 2014) and content filtering based on user/item features (Rendle, 2012). Collaborative filtering techniques that use user-item interaction histories are often limited by the model capacity in their ability to embed, summarize and use the ordered sequences of long histories. This observation has driven many model improvements ranging from multi-hot auto-encoders (Sedhain et al., 2015) to n-gram factorization machines (Rendle, 2012) and recurrent neural networks (RNNs) (Hidasi et al., 2015) for recommendations.

One important limitation when adopting the basic RNNs from language modeling, e.g. Cho et al. (2014) for recommendations is their limited ability to model long sequences due to vanishing gradients. While natural sentences have a typical length of twenty words, a user browsing a website easily receives twenty recommendations in a few minutes and consumes more than twenty items each day. To solve this problem, we group user activities into sessions, where we assume users keep the same interests. These sessions can be identified by simple heuristics, e.g. thirty minutes of inactivity.

Workshop on Modeling and Decision-Making in the Spatiotemporal Domain, 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

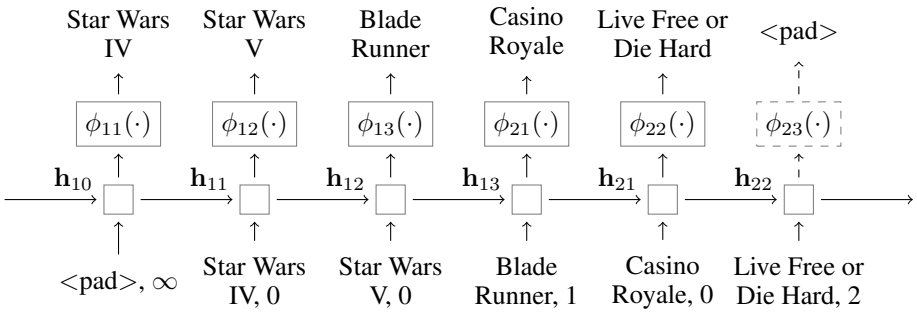

Figure 1: HRNN sequence model to predict the next item in a recurrent fashion. The second input position encodes a categorical hierarchy control signal which can be inferred from time-deltas. HRNN may be combined with temporal convolution networks (Bai et al., 2018) to model long sequences.

Quadrana et al. (2017) further proposed a two-layer hierarchical GRU (HGRU) model: one layer for inter-session dynamics and another layer, which performs updates only at the end of each session, for cross-session dynamics. As a result, HGRU can more easily learn from long sequences due to short gradient paths between more distant pasts and futures. However, direct manipulation of the RNN hidden states breaks truncated back-propagation through time (TBPTT, Williams & Peng (1990); Merity et al. (2017)) training paradigm, where multiple user histories are concatenated to a long sequence and then split at even intervals for back-propagation efficiency. In this paper, we encode the session information at the inputs, through a learned embedding, which not only achieves computational efficiency but also allows us to easily extend these hierarchical structures to other RNNs (e.g., LSTMs) and beyond (e.g., Transformers). We call our model HRNNs. Related to our work, Li et al. (2017) model time-deltas, but without making the connection to hierarchical sessions.

The second contribution of our work is to show to include meta data such as user and item features, directly into sequence models. The basic idea is to score each user and item pair using their separate features, where some subsets of the features are learned representations from the RNN, followed by a ranking loss. To focus on the interaction effects between user and item features, Rendle (2012) explicitly included higher-order multiplicative terms in the scoring function. We extend the observation to sequence models, by considering second-order terms between user embedding vectors from the RNN and item embedding vectors from a separate feed-forward network. The resulting models include vanilla RNNs as a special case, if we view the RNN decoder weights as embedding vectors from one-hot encoded item ids.

2 Hierarchical Recurrent Networks with Meta Data (HRNN-Meta)

We start with the problem of making personalized recommendations based on an interaction dataset, where each row contains a historical record of {(time, user id, item id, value)} $= \{(t_k, u_k, a_k, v_k) : k = 1, \ldots, n\}$. Value can be used to represent click vs. no-click or click vs. purchase. We will further extend the representation to include other interaction labels (e.g. rating, purchase value, no click), user profiles, product features, etc.

Sequence models gain statistical power by aggregating "people who watched X also watched" from individual user histories, while personalizing to recent trends in customer behaviour. For this purpose, we group the interactions into ordered user histories, $\mathcal{X}_i = [x_{i1}, x_{i2}, \ldots, x_{ik_i}]$, with temporal re-indexing, where $x_{ik} = (a_k, t_k, v_k)$. We focus on single user histories and omit the user indices below, noting that we learn the global item-to-item dynamics by stochastic gradient descent on individual user histories. We use RNNs such as GRUs to predict the probability of each item that a user may watch next in their watch history, similar to how language models predict the next word in a sentence (Zaremba et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014). Let A_{k+1} be the random variable representing the next watch item a_{k+1}, $x_{1:k} = (x_1, \ldots, x_k)$ be the relative history, and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)^\top$ be the scoring vector function corresponding to all items; we model:

$$a_{k+1} \sim p(A_{k+1} \mid x_{1:k}) = \mathrm{softmax}(\boldsymbol{\phi}(\mathbf{h}(x_1, \ldots x_k))), \tag{1}$$

where $\mathbf{h}_k = \mathbf{h}(x_{1:k})$ is a hidden state representation of the respective user history. A naive state can be the multi-hot encoding of past items, i.e., a sparse vector with each nonzero location correspond

to the index of a past item. If we use $\boldsymbol{a}_k = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ to denote the indicator vector corresponding to the item index a_k, the hidden state is then $\mathbf{h}(x_{1:k}) = \boldsymbol{a}_1 + \cdots + \boldsymbol{a}_k$. In practice, RNNs can be viewed as a weighted sum of representations of past events, where the weights are 'gated' by the salience of every item in the context of items the customer has interacted with so far.

Figure 1 sketches the RNN sequence models and our hierarchical extensions. While equation 1 describes the prediction of every single event, actual training back-propagates gradients through the entire sequence (up to a fixed maximum length called TBPTT interval) in a single batch.

In our model, **hierarchical sessions** are used to divide long user histories into short activity intervals within which the sequence has some consistency, e.g. a single user intent. Session breaks are often inferred from periods of inactivity. Instead of direct manipulation of GRU hidden states as in HGRU (Quadrana et al., 2017), which limits model expressiveness and computational efficiency, we simply concatenate session start 'control signals' with the HRNN inputs, shown as the second input bit in Figure 1. Our experiments empirically verify that allocating special "control" inputs allows the model to reset states as appropriate in properly trained HRNNs. In addition to having comparable results, the "control" channels allow the model to accomplish complex tasks, e.g., to retrieve an earlier item from the previous session and copy it to proper locations in the next or future sessions.

Item features can be naturally included in HRNNs, if we notice the connections between RNN decoders and factorization models. Vanilla RNNs find the score of the jth item with function $\phi_j(\mathbf{h}_k) = \mathbf{w}_j^\top \mathbf{h}_k + b_j$, where $\mathbf{w}_j \in \mathbb{R}^r$ is the loading coefficients and $b_j \in \mathbb{R}$ comes from popularity bias. We view $\mathbf{w}_j$ as the embedding vector of the jth item and consider the decoder as a second-order interaction model between $\mathbf{x}_j$ and $\mathbf{h}_k$. A natural extension:

$$\tilde{\phi}_j(\mathbf{h}_k) = (1-\lambda)\phi_j(\mathbf{h}_k) + \lambda\big(\mathbf{w}(\mathbf{f}_j)^\top \mathbf{h}_k + b(\mathbf{f}_j)\big),$$

where $\mathbf{w}(\cdot), b(\cdot)$ are learnable functions that embed item features and $0 \le \lambda \le 1$ is a mixing parameter. Item features are particularly useful when recommending **cold-start** items where no interaction data is available. In this scenario, we found that item feature embeddings should be learned as substitutes rather than complements of the decoder weights. We achieve this by randomizing λ during training.

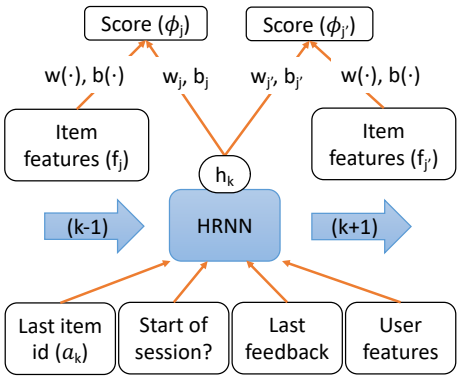

Figure 2: HRNN-meta cell model.

Incorporating **user features** (e.g. location) and **interaction feedback** (e.g. rating, purchase cost) often have significant impact on the recommendation quality. We concatenate learned dimension-reduced embedding representations of both with the RNN inputs, i.e., similar to session control signals (Figure 2). More complex models such as negating the RNN inputs for negative feedback or deep-cross models (Wang et al., 2017) yielded similar performance in our preliminary experiments.

3 Experiments

We conducted ablation studies with both simulation and real data to show the effects of every aspect of HRNN-meta, including session information, user and item features, and interaction feedback. As a test dataset we used MovieLens (Harper & Konstan, 2016), a dataset of movie ratings. The conclusions were largely insensitive to specific hyper-parameters. A good reference is Gluon-NLP. [1]

Table 1: Simulation with hierarchical sessions

	RNN	HGRU	HRNN
session-aware	×	✓	✓
model hierarchies	1	2	1
test PPL	2	1	1
time per epoch	0.3s	25.5s	0.3s

Session information was shown directly in simulation studies and also indirectly in real-data experiments. We simulated sequences containing random numbers (0-5) of sessions, each of which contains a random number (1-5) of items. The item ids remain the same within a session - to reflect similar user interests - and increment by 1 between sessions.

[1] https://github.com/dmlc/gluon-nlp/blob/master/scripts/language_model/index.rst

Table 1 compared three models: a vanilla RNN, a HRNN that encodes sessions as inputs, and a HGRU (Quadrana et al., 2017) that manipulates hidden states with session data. All models were trained to predict all items and evaluated only for the first item of each new session. Particularly, both HRNN and HGRU have a separate input channel for session-start signals, which are then handled differently (Section 2). We used perplexity (PPL, Brown et al. (1992)) to measure the model inaccuracy. A PPL of p is equivalent to a recommender that performs a uniform random selection from p items, one of which is the true next item. The vanilla RNN is not session-aware and was often incorrect about the next item. It did learn the average session length, and hence was able to predict correctly with 50% chance. Both HGRU and HRNN are session-aware and predicted the correct first in session item. HRNN has significant reduced training time due to temporal alignment in mini-batches.

We evaluated **Sequential meta data models** with a toy simulation: The recommender chooses an item to bet on (i.e. a measure of the model's confidence level) at each time. The first item is uniformly random, and is also presented as user feature side information. So a model that correctly uses user features, should be able to predict the first item, since it's id is leaked. The following bets are always +1 or -1 in item ids depending on the previous interaction outcomes. The parity of the bet is recorded as item features. The total range is 0-100 and there are 1000 independent sequence of random lengths 0-20.

Table 2: Meta data PPL benchmarks

Seq	User	Item	Feedback	Toy	ml-20m
×	×	×	×	100	2228
×	×	✓	×	54	1342
✓	×	×	×	3	447
✓	✓	✓	✓	1	410

Table 2 shows that besides interaction outcomes, all modeling aspects were useful. The user features reveal the first bet, with which the gambler's bets can be narrowed to an average of 11 items. Using sequence models without knowing the first item will only have 1-in-3 chance of success, including the first item. Having full information allows us to make perfectly accurate predictions.

We use **movielens data (ml-20m)** as a real-world dataset for movie recommendations. It contains 20 million interactions, 131 263 items,[2] and 138 493 unique users. We split the data by user ids into 80% train and validation set and 20% test set. Table 2 shows that our vanilla RNN implementation achieved 3x the performance of popularity baseline, i.e., PPL 447 versus 2228, similar to (Donkers et al., 2017).[3] The meta data model also improved the prediction accuracy, driving PPL to 429 and 410. Here, item features come from movie genre vectors, with L^2 normalization; interaction feedback were the standardized rating values; and user features were faked by using the features of the first item. Notice, item features also improved popularity baseline to PPL 1342, using a vanilla factorization machine (Rendle, 2012) with features from a user's last item and features of all possible future items.

Since the dataset has both positive and negative feedback, we can extend the objective to **root-mean-square error (RMSE) for rating predictions**. Absolute values across papers are not meaningful[3], but we did observe improvements as the model complexity increases. One particular observation is that RNNs without feedback encoding are unaware of the user average ratings and could not outperform the rolling average baseline - which predicts the next item rating as a simple average of the item average rating and user average rating. RNN-meta overcame this limitation. HRNN-meta can use extra contextual information at inference time (e.g. to trade-off relevance and popularity depending on time since last customer interaction - see appendix).

Table 3: Ml-20m rating prediction.

	RMSE
Rolling average baseline	0.933
Factorization (Rendle, 2012)	0.916
I-AutoRec (Sedhain et al., 2015)	0.871
RNN wo. feedback encoding	0.941
RNN w/ feedback encoding	0.857
HRNN w/ feedback encoding	0.846

4 Conclusions

We extend recurrent recommender systems with hierarchical session information and various meta data types. The improvements were demonstrated with both intuitive simulations and real-world data.

[2]We treat the size of the index space as the number of items despite only 26 744 unique items being included.

[3]There are no standard temporal train-test splits on ml-20m, so we cannot make a more direct comparison.

Acknowledgement

We greatly appreciate helps from AWS AI and AWS Personalize teams. We especially thank Haibin Lin, Chenguang Wang, and Aston Zhang for their detailed discussions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Peter F Brown, Vincent J Della Pietra, Robert L Mercer, Stephen A Della Pietra, and Jennifer C Lai. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 152–160. ACM, 2017.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Yang Li, Nan Du, and Samy Bengio. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065*, 2017.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 130–137. ACM, 2017.

Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 111–112. ACM, 2015.

Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pp. 12. ACM, 2017.

Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

A Example on session and intent

Users tend to keep their intents in-session and and change intents between sessions. The example below shows that the learned HRNN models tend to follow a similar pattern. For example, based on item watch history reflected in Table 4, HRNN-meta generates different recommendations for time-delta gaps corresponding to different session-start hierarchies (controls) (Table 5). When the recommendation is made by HRNN within session, that is, when a short time delta is used at query time, the theme stays relatively similar. As session control indicates a larger hierarchy, the recommendation genres tend to be more diverse, i.e., move toward the base popularity with diminishing personalization effects, as desired. We observed better performance when the hierarchy is neither too short (leading to greedy repetitive choices) or too large.

Table 4: Example user watch history with relative time in seconds

time	title	genres	popularity
-221	Secret of Roan Inish	Children|Drama|Fantasy|Mystery	0.000258
-185	Postman	Comedy|Drama|Romance	0.000718
-146	Thin Blue Line	Documentary	0.000153
-99	Say Anything...	Comedy|Drama|Romance	0.000493
0	Babe: Pig in the City (1998)	Children|Comedy	0.000287

Table 5: Top-1 recommendation to the same user changes as the inference time changes.

time	hierarchy	title	genres	popularity
0	0	Purple Rose of Cairo	Comedy|Drama|Fantasy|Romance	0.000236
60	1	Unbearable Lightness	Drama	0.000209
3600	2	Local Hero	Comedy	0.000195
86400	3	Big	Comedy|Drama|Fantasy|Romance	0.001130