Temporal Learning and Sequence Modeling for a Job Recommender System

Kuan Liu[1], Xing Shi[1], Anoop Kumar[2], Linhong Zhu[2], and Prem Natarajan[1]
[1,2]Information Sciences Institute, [1]Computer Science Department, [1,2]University of Southern California
{liukuan,xingshi,anoopk,linhong,pnataraj}@isi.edu

ABSTRACT

We present our solution to the job recommendation task for *RecSys Challenge 2016*. The main contribution of our work is to combine temporal learning with sequence modeling to capture complex user-item activity patterns to improve job recommendations. First, we propose a time-based ranking model applied to historical observations and a hybrid matrix factorization over time reweighted interactions. Second, we exploit sequence properties in user-items activities and develop a RNN-based recommendation model. Our solution achieved 5^{th} place in the challenge among more than 100 participants. Notably, the strong performance of our RNN approach shows a promising new direction in employing sequence modeling for recommendation systems.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommendation systems, Recsys Challenge, Temporal, Sequence, LSTMs, Recurrent Neural Networks

1. INTRODUCTION

The problem of matching job seekers to postings [6] has attracted lots of attention from both academia and industry (e.g., Xing[1] and LinkedIn) in recent years. *Recsys Challenge 2016* is organized around a particular flavor of this problem. Given the profile of users, job postings (items), and their interaction history on Xing, the goal is to predict a ranked list of items of interest to a user.

To develop a high quality job recommendation system, one needs to understand and characterize the individual profile and behaviors of users, items, and their interactions. The commonly used factor models [3] learn factors for user and item by decomposing user-item interaction matrices. Neighborhood methods [7] rely on similarities between users and items that are derived from content or co-occurrence. These popular methods often ignore or underexploit important temporal dynamics and sequence properties between users and items.

To address the aforementioned limitations, we explore temporal and sequence modeling to characterize both temporal behaviors and content similarity of users and items. First, we propose a time-based ranking model that leverages historical interactions for item recommendation. Second, we investigate how to learn latent temporal factors from both user-item interactions and their associated features. Instead of factoring a single aggregated matrix, we extend the context-aware matrix factorization model to explicitly consider temporal interactions.

Finally, motivated by recent success of sequence modeling [1, 2, 8], we explore the Recurrent Neural Networks (RNNs) approach to capture user-items behavior patterns. We suggest that sequence modeling is very helpful in terms of modeling both item-item similarity and item temporal transition patterns and it consequently leads to a more effective way of utilizing item history. Towards this end, we develop an Encoder-Decoder sequence recommendation system that incorporates feature learning, which significantly increases model flexibility.

The contributions of this work are summarized as follows: 1) A novel temporal ranking approach to recommend items from history; 2) An enhanced hybrid matrix factorization model that explicitly incorporating temporal information; 3) A RNN-based sequence model that considerably outperforms matrix factorization models; and 4) Our final system, an ensemble of the above components, achieving 5^{th} place in *RecSys Challenge 2016*.

2. PROBLEM AND DATA: *RECSYS 2016*

RecSys Challenge 2016 provides 16 weeks of interactions data for a subset of users and job items from the social networking and job search website - *Xing.com*. The task is to predict the items that a set of target users will positively interact with (click, bookmark or reply) in the following week.

Data Set. Users and items are described by a rich set of categorical or numerical features or descriptor features. Categorical features take several to dozens of values and descriptor features have a vocabulary size around $100K$. Observation including positive interactions and impressions (items shown to users by *Xing*'s existing recommendation system) at different weeks are also available. The detailed quantitative information of this dataset is shown in Table 1.

Task and Evaluation Metric. Given a user, the goal of this challenge is to predict a ranked list of items from the active item set. The score is a sum over scores of each user $S(u)$, which is defined as follows:

$$\begin{aligned}\mathbf{S}(u) =& 20 * (P@2 + P@4 + R + \textsc{UserSuccess}) \\ &+ 10 * (P@6 + P@20)\end{aligned}$$

[1]https://www.xing.com/

RecSys Challenge '16, September 15 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2987538.2987540

Table 1: Statistics of Dataset and Feature Description.

Data splits	Sizes
Target/all users	150K/1.5M
Active/all items	327K/1.3M
Interactions	8.8M
Impressions	202M
Feature types	**Features**
Categorical (U)	career_lever, discipline_id, industry_id, id, country, region, exp_years, exp_in_entries_class, exp_in_current
Descriptors (U)	job_roles, field_of_studies
Categorical (I)	id, career_level, discipline_id, country, region, employment
Numerical (I)	latitude, longitude, created_at
Descriptor (I)	title, tags

where $P@N$ denotes the precision at N, R is the recall, and USER-SUCCESS equals 1 if there is at least one item correctly predicted for that user.

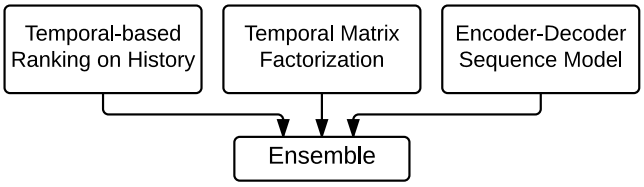


Figure 1: System Overview.

3. METHODS

The proposed solution consists of three main components (see Fig. 1). We describe each of these in the following.

3.1 Temporal-based Ranking on History Items

The first component of our system is motivated by the observation that *users have a strong tendency to re-interact with items that they already did in the past*. Statistically, on average 2 out of 7 items from the first 14 weeks re-appear in the 15^{th} week's interaction list. It makes it very plausible to recommend the old items to users. Similarly, items that appeared in the "impression" list are also preferred. It motivates us to consider the set of items from past interactions and impressions as a candidate set for user.

Given a user u and an item i, the historical interactions between u and i before time t is represented as $M_{i,u,t} \in \mathbb{N}^{K \times T}$, where T is number of time stamps from time 1 to time t, K is number of types of interactions (e.g., click, bookmark, or reply), and $M_{i,u,t}(k,\tau)$ is the number of k-type interactions at time $\tau \in \{1, \cdots, t\}$. Given a user u, a naïve model is to rank each item simply based on the aggregation adoption of history, which leads to

$$S(u,i,t) = \sum_k \sum_{\tau=1}^{t} M_{i,u,t}(k,\tau)$$

where $S(u,i,t)$ evaluates how likely user u is to re-interact with an item i given their historical interactions.

However, not every historical interaction by a user has the same importance. For example, a user may prefer re-clicking an item from previous day over one clicked 10 weeks ago. We conjecture that the importance of user-item interactions depends on the time of interaction. With this assumption, given a user u, item i and a particular time t, we propose a time reweighed linear ranking model, which is defined as:

$$S(u,i,t) = wM_{u,i,t}^T$$

where w is the coefficient associated with time, with $w(k,\tau)$ indicating the relative contribution of k-type interactions at time τ.

To learn w, we construct triplet constraints

$$\mathcal{T} = \{u \text{ prefers to re-interacting with } i_1 \text{ to } i_2 \text{ at time } \tau\}_{n=1}^{N}.$$

when u *interacted with* i_1, i_2 *before* τ, *but only interacted with* i_1 *at* τ. We thus obtain the solution of w by minimizing an objective function that incurs a smoothed hinge loss when a constraint is violated.

3.2 Factorizing Temporal Interactions

3.2.1 Hybrid Matrix Factorization and Categorical Feature Learning

In order to recommend items that one user interacted with to another similar user or to recommend newly appearing items, we exploit the availability of user/item features, and characterize items/users by vectors of latent factors inferred from their features.

Our approach starts with hybrid matrix factorization technique [5]. To briefly review, we model each user/item as a sum of the representations of its associated features and learn a d-dimensional representation for each feature value (together with a 1-dimensional bias). Let $\vec{x}_j^U/\vec{x}_j^I$ denote the embedding (i.e., vectors of factors) of the user/item feature j, $\vec{q}_u/\vec{q}_i$ denote the embedding of user u /item i, and b_j^U/b_j^I denote the user/item bias for feature j. Then

$$\vec{q}_u = \sum_{j \in f_u} \vec{x}_j^U, \vec{q}_i = \sum_{j \in f_i} \vec{x}_j^I; \qquad b_u = \sum_{j \in f_u} b_j^U, b_i = \sum_{j \in f_i} b_j^I \tag{1}$$

The model prediction score for pair $\{u, i\}$ is then given by

$$S(u,i) = \vec{q}_u \cdot \vec{q}_i + b_u + b_i \tag{2}$$

The model is trained by minimizing the sum of a loss on $S(u,i)$ and the observed ground truth $t(u,i)$,

$$L = \sum_{\{u,i\} \in I} \ell(S(u,i), t(u,i)) \tag{3}$$

where I is set of interactions between user u and item i, ℓ is chosen to be Weighted Approximately Ranked Pairwise (WARP) loss.

3.2.2 Temporal re-weighted Matrix Factorization

Our approach is grounded on the assumption that the time factor plays an important role in determining the user's future preference. To this end, we place a non-negative weight associated with time on the loss, which leads to the following equation:

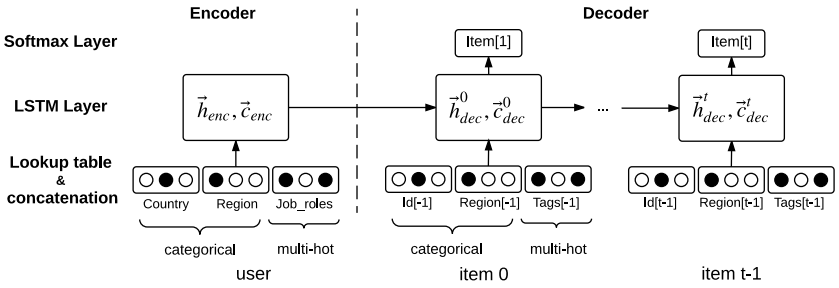
$$L' = \sum_{\{u,i,\tau\} \in I} \ell(S(u,i), t(u,i,\tau)) \times \gamma(\tau) \tag{4}$$

Here the re-weighting term γ depends on the time τ when the user-item interaction happens, which captures the contribution from interactions over time. Additionally, some zero weight γ reduces training set size to speed up training and could possibly help prevent over-fitting.

In general γ can be learned jointly with other embedding parameters in the model. In practice, we only fixed γ as the learned weights w from Model 3.1 to speed up training.

3.3 Sequence Modeling via RNNs

In this section, instead of viewing user-item interactions as independent pairs, we model the entire set of user-item interactions from the same user as a sequence ordered by time. Sequence modeling may reveal the sequential patterns in user-item interactions such as the shifting of user interests over time and the demanding evolving of job items.



Encode u,

$$\vec{h}_{enc}, \vec{c}_{enc} = LSTM(f(u), \vec{0}, \vec{0})$$

Update $\vec{h}_{dec}, \vec{c}_{dec}$,

$$\vec{h}^t_{dec}, \vec{c}^t_{dec} = LSTM(f(i^{t-1}), \vec{h}^{t-1}_{dec}, \vec{h}^{t-1}_{dec})$$

Figure 2: Encoder-Decoder model for recommendation.

3.3.1 Encoder-Decoder modeling

We develop an Encoder-Decoder model [8] based on LSTM shown in Fig 2. Given a user u and her interaction item sequence $\mathtt{I} = \{i^1, ..., i^T\}$, u is encoded into vectors $\vec{h}_{enc}$ and $\vec{c}_{enc}$; At decoding phase time step t, $\vec{h}^t_{dec}$ and $\vec{c}^t_{dec}$ are updated and $\vec{h}^t_{dec}$ is used to predict i^t. i^0 here is a special START item. Cross Entropy is used as the training loss.

3.3.2 Novel Extensions

Features. f in Fig 2 is a function that maps a user/item index (profile) into a vector by concatenating its features' embedding. For categorical features, the embedding is extracted from a look-up table, and descriptor features are considered as multi-hot features and average pooling is used. The look-up table is jointly learned during training.

Anonymous users. Item IDs are used as categorical features to capture item characteristics that are beyond feature descriptions. However, we remove user IDs from user feature set to prevent overfitting to those IDs and empirically observe better performance. It also leads to our natural train/validation set split by randomly splitting user set.

Sampling and data augmentation. Unlike the common success of data augmentation [4] and existing item sampling techniques [9], our results indicate that it is better to use the original sequences, *without sampling items*, to construct training set. Results and analysis are reported in Section 4.2.

4. EXPERIMENTS

4.1 Settings

We take user-item interaction data from the 26^{th} to the 44^{th} weeks as training data and validate our model on the 45^{th} week. Submitted results come from models re-trained on data from 26^{th} to 45^{th} week under the same hyper-parameters. We observe very strong correlation between validation and test scores for all our models and thus **mostly report validation scores** below due to submission quota.

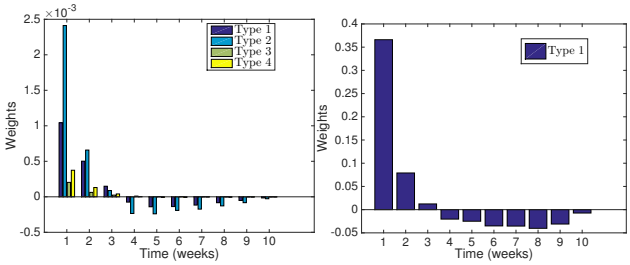
4.2 Results

4.2.1 Recommend from History

Our model in Section 3.1 (TRank) is compared to two baseline models: randomized score (Rand) and recency-based sorting (TSort) that sorts items by the latest time they appear in the history. The results on historical "interactions"(INTS), "impressions"(IMPS), and their combinations (INTS+IMPS) are reported in Table 2. TRank clearly outperforms the other two in all the three cases. Figure 3 shows the learned weights w associated with the designed temporal features. The coefficients are decaying with time in both Figure 3(a) and 3(b) across different types of interactions,

Table 2: Scores in thousands (K) based on history interactions.

Models	Rand	TSort	TRank
INTS	266	284	**299**
IMPS	324	375	**380**
INTS+IMPS	463	509	**524**



(a) interactions (b) impressions

Figure 3: Weights learned in Model 3.1 for interactions 3(a) and impressions 3(b). K=4 for INTS and K=1 for IMPS, where type 1 denotes user-item impression pairs, and type 2,3,4 denote click, bookmark and reply, resp.

indicating that more recent interactions have a larger statistical impact over the users' future preferences. Furthermore, although recency is important, simply using the latest time performs worse than TRank, which smoothly combines the most recent interactions with historical interactions using the learned weights w.

4.2.2 Recommend via Matrix Factorization

We compare hybrid matrix factorization model in 3.2.1 (HMF) and the model in 3.2.2 (THMF) with different number of latent factor d, without and with features. Two important measures are used: score$_{all}$ and score$_{new}$. score$_{all}$ is the challenge score and score$_{new}$ is the score after removing all history user-item pairs. We found score$_{new}$ more important in model ensemble and chose in our experiments to early stop model training at the best score$_{new}$.

As shown in Table 3, THMF models achieve significant improvements on score$_{all}$ and score$_{new}$ for all d, with and without features. Meanwhile, the time comparison shows that the best models achieved by THMF require significantly less training time.

Finally, we use items in the "impression" list in last week and treat them as "interactions" (with 0.01 down-weight). This boosts performance as seen in Table 4.

4.2.3 Recommend via LSTMs

Setting, Tuning, Details. With extensive tuning, we choose to train the encoder-decoder model (LSTM) using a single layer LSTM with hidden vector size as 256 and dropout rate as 0.6. All parameters are uniformly initialized before $[-0.08, 0.08]$. The learning rate of Stochastic Gradient Descent (SGD) is initially set as 1.0 and will decay with rate 0.7 once the perplexity on development set starts

Table 3: Scores (K) achieved by hybrid matrix factorization models and training time (in hours h) .

Models		HMF			THMF		
Fea	d	score$_{all}$	score$_{new}$	T	score$_{all}$	score$_{new}$	T
No	16	235	61	8.8	269	65	2.8
	32	301	71	3.4	320	75	1.5
	48	313	78	7.7	326	84	1.7
	64	330	76	3.3	340	86	0.7
Yes	16	311	124	74	361	146	34
	32	326	125	26	**381**	**148**	14
	48	354	128	76	378	144	12

Table 4: Scores (K) by THMF with some "impression"s as additional observation inputs.

Observations	INTS	INTS + IMPS
score$_{all}$	381	**438**
score$_{new}$	148	**164**

to increase. We only consider the top 50,000 frequent item and replace the remaining item as UNK. The final result is ensembled using 6 models with different random seeds.

Performance. Table 5 reports the comparison between HMF, THMF and LSTM. Models are trained on the datasets[2] with and without features. Provided with features, LSTM obtains superior results to the rest. It verifies that sequence modeling is a promising direction for job recommendation tasks.

We also note when features are not provided, LSTM does not show advantages to THMF. We don't know the exact reason yet but suspect it is due to the inflexibility of non-feature sequence model which has a hard time capturing item transition pattern.

Table 5: Scores (K) comparison among HMF, THMF, and LSTM models. All models are trained on active item set.

Fea	No			Yes		
Models	HMF	THMF	LSTM	HMF	THMF	LSTM
score$_{all}$	313	**347**	313	312	366	**391**
score$_{new}$	78	87	**89**	104	130	**140**

Does sequence help? When applying sequence modeling to the recommendation problem, we implicitly assume that sequence or order provides additional information beyond that provided by item frequency alone. To test the validity of this assumption, we generated new training data through sampling sub-sequences in which items were dropped out with certain probability. At the same time, on the average, item appearance frequency would remain unchanged in data with more sampled sub-sequences.

Experimental results with the new generated training data are shown in Fig 4. First, increasing sub-sequence sampling leads to decreasing scores (from x_1 to x_8); Second, original data set (full sequences) gives the best score. These results suggest that item sequences do indeed provide additional information and merit further investigation.

4.2.4 Ensemble and Final Results

Given the prediction and confidence scores from the above model components, we perform model ensemble to obtain our final result. The final scores are achieved by using a Random-Forests-based ensemble approach, which outperforms the linear fusion by greedy grid search in our experiment. Final scores for different components and the ensemble model are reported in Table 6.

[2]To make fair comparison, all models are trained on full user set and active item set. HMF, THMF do benefit from training with additional items (see Table 3); however, currently we don't have efficient implementation of LSTM supporting additional items.

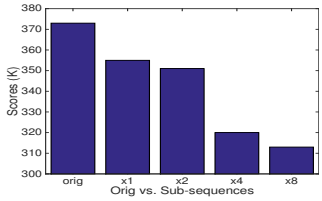


Figure 4: LSTM scores (K) on original and "manipulated" data sets. "Orig" denotes the complete sequence. x_N denotes the manipulated data set obtained by randomly sampling subsequence proportional to N times.

Table 6: Final Component and Ensemble Results.

Component	History	MF (ints+imps)	LSTMs	Ensemble
Valid	524	438	391	613
Test	502	441	384	615

5. CONCLUSIONS

In this paper, we presented our innovative combination of new and existing recommendation techniques for *RecSys Challenge 2016*. Empirical study verified the effectiveness of 1) utilizing historical information in predicting users' preferences and 2) both temporal learning and sequence modeling in improving recommendation.

Notably, the proposed RNN-based model outperforms the commonly used matrix factorization models. In the future, we would like to extend our research in model designs (e.g. to incorporate features in the output layer and to support other loss functions) and in result analysis to understand why and when the sequence modeling really helps recommendation.

6. ACKNOWLEDGMENTS

This paper is based upon work supported by the DARPA DEFT Program. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

7. REFERENCES

[1] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[2] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[3] Y. Koren, R. Bell, C. Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] M. Kula. Metadata Embeddings for User and Item Cold-start Recommendations. *arXiv preprint arXiv:1507.08439*, 2015.

[6] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel. Matching people and jobs: A bilateral recommendation approach. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 137c–137c, 2006.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[8] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[9] Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. *arXiv preprint arXiv:1606.08117*, 2016.