Latent Trajectory Modeling: A Light and Efficient Way to Introduce Time in Recommender Systems

Elie Guardia-Sebaoun
Sorbonne Universites,
UPMC, LIP6, UMR 7606,
4 place Jussieu,
F-75005 Paris, France
elie.guardia-sebaoun@lip6.fr

Vincent Guigue
Sorbonne Universites,
UPMC, LIP6, UMR 7606,
4 place Jussieu,
F-75005 Paris, France
vincent.guigue@lip6.fr

Patrick Gallinari
Sorbonne Universites,
UPMC, LIP6, UMR 7606,
4 place Jussieu,
F-75005 Paris, France
patrick.gallinari@lip6.fr

ABSTRACT

For recommender systems, time is often an important source of information but it is also a complex dimension to apprehend. We propose here to learn item and user representations such that any timely ordered sequence of items selected by a user will be represented as a trajectory of the user in a representation space. This allows us to rank new items for this user. We then enrich the item and user representations in order to perform rating prediction using a classical matrix factorization scheme. We demonstrate the interest of our approach regarding both item ranking and rating prediction on a series of classical benchmarks.

1. INTRODUCTION

During the last decade, the emergence of collaborative filtering has demonstrated the interest of exploiting past ratings to establish relevant profiles for both users and items in various application fields [17, 10]. Recent works has focused on profile enrichment: the better we understand users' tastes, the more accurate our suggestions will be. Several directions have been investigated for this: global ratings of reviews have been manually split into different aspects [7], then [11] has proposed to automate this process by learning a common latent space to represent the items, the users and the reviews they wrote. Yet another approach directly exploits tokens extracted from those reviews in order to improve user characterization [15]. Time is another important dimension since the perception of a user may change with time or be time dependent. There are different ways to handle time, for example one can consider the temporal evolution of items perception [9] or model the evolution of the user's way of thinking [12]. Considering the time dimension usually considerably increases the complexity of recommendation models. Instead of time, we propose to consider the ordering of user actions and to embed this information into the users and the items representations for recommendation tasks. Ordered sequences convey less information than full time sequences of user actions, but allow for a compromise between the information provided to the system and the system complexity needed to handle this information. More precisely, we consider ordered sequences of items corresponding to sequences of user actions. We learn a representation -a vector in a multidimensional euclidean space- of individual items in the context of the sequences where they appear. Each item will then have a unique vector representation in a vector space. A user representation will be a function operating on this item representation space that allows to move from one item to the next one, or said otherwise to infer the next item from the current one. In this paper, we consider a simple vectorial operator corresponding to a translation. Given a sequence of items, this allows us to infer the future sequence and then to recommend at each step an ordered list of items to a user. We then enrich this representation by additional dimensions and use a classical matrix factorization scheme in order to learn this new representation for learning to predict item ratings like in classical collaborative filtering.

This approach offers two advantages: it models the dynamical aspects of the user profile while keeping the number of parameters low. We demonstrate the effectiveness of our approach on classical review datasets through two evaluation schemes: item ranking and rating prediction.

The paper is structured as follows: related work is reviewed in Section 2, then we describe the model and the training algorithm in Section 3. In Section 4, we describe the datasets and our results.

RecSys'15, September 16–20, 2015, Vienna, Austria.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3692-5/15/09…$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2799676.

2. RELATED WORK

Recommender systems can be evaluated in two different setups, one consists in recommending a set of items (ranking) [3, 13] while the other aims at estimating how a user would rate a given item (rating prediction) [17, 2, 8].

In the recommendation literature, time is often considered as a relevant contextual information as it facilitates detecting the changes in users preferences [18]. A better understanding of these changes allow the recommender systems to capture periodicity in users interests [5] or simply to improve its performance [9].

However, capturing temporal dynamics in a recommender engine is not straightforward. For example using an exponential decay to downweight older reviews either improve [6] or degrade [9] the system performance. As another example, while in [1], the authors used a time-dependent data partition to learn context-aware user profiles, they found best results for a random split. [4] surveys different ways to handle time inthe recommender literature like using temporal drift, heuristics or splits. Two recent interesting contributions for introducing time in recommender systems are [19] who uses Kalman filters to represent a transition between the latent representations of the users over time and [12] who represents this evolution as user experience gained over time.

3. CONTRIBUTION

We will use the following notations: Items (resp. users) are indexed using i (resp. u) and gathered in a set: $I = \{i_k\}_{k=1,\dots,N}$ (resp. $U = \{u_k\}_{k=1,\dots,M}$). To each user corresponds a trace θ_u, *i.e.* an ordered sequence of items he rated: $\theta_u = \{(i,r)|i \in I\}$ where r is the rating ; those traces are gathered in a set $\Theta = \{\theta_u | u \in U\}$.

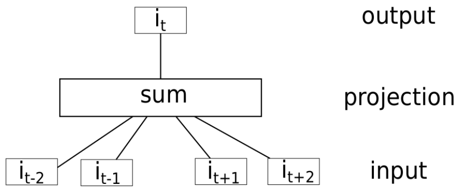

Figure 1: The *Mikolov et al.* [14] architecture predicts the current item based on the context

Item Latent Representation

Mikolov et al. Word2Vec model [14], as illustrated in Fig.1, uses a neural network to build a latent representation of words depending on the surrounding words in a sentence, by optimizing a word prediction criterion. In this paper we used the same neural net architecture for learning from item sequences corresponding to user traces $\{\theta_u, u \in U\}$, a contextualized item representations. For every item i, we build a representation ϕ_i (in blue in Fig.2) that considers the items order in all the sequences corresponding to user traces.

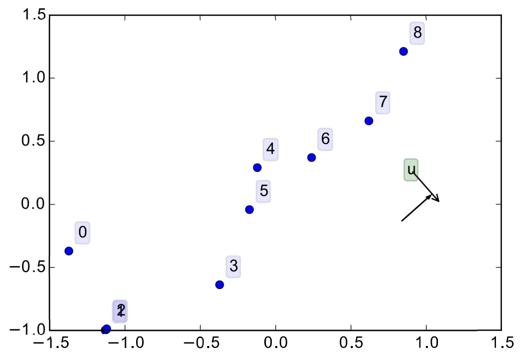

Figure 2: Representation for d=2 of an extract of a user trace (numbered items in blue) and the user representation as a vector (in green) for the RateBeer dataset.

Ranking Function Optimisation

For every user u, we learned a representation ϕ_u (u in Fig. 2) modeling the way u moves from an item to the next one in the sequence representation in the latent space, by performing a gradient descent on the following ranking loss:

$$\mathcal{L}_{ranking}(\Theta) = \sum_{u \in U} \sum_{i \in \theta_u} \sum_{j \in I \setminus x_n} \left[1 - |\phi_u + \phi_{x_n} - \phi_i|^2 - |\phi_u + \phi_{x_n} - \phi_j|^2\right]^+ \quad (1)$$

In equation 1, the $^+$ sign indicates that we used a hinge loss function. This means that during the learning phase, we only updated the parameter values when $\mathcal{L}_{ranking}(u,i,e) > 0$.

ϕ_u is a vector which represents the mean of the translations needed to move from one item in the representation space to the next one in a sequence corresponding to a user trace. From these representation, we can - by applying the ϕ_u translation from an item representation- compute the nearest neighbors of the resulting point and thus produce ordered list of items to recommend. Given a user u and i the most recent item he reviewed, the item recommendation is given by:

$$rec(u,i) = argmin_{j \in I} \left\|(\phi_i + \phi_u) - \phi_j\right\| \quad (2)$$

Rating Function Optimisation

In a second step, we construct new representations γ_u and γ_i by enriching ϕ_u and ϕ_i, in order for our model to tackle the rating task.

- $\gamma_u = [\bar{\gamma}_u, \phi_u], \forall u \in U$
- $\gamma_i = [\phi_i, \bar{\gamma}_i], \forall i \in I$

Where $\{\bar{\gamma}_u, u \in U\}$ and $\{\bar{\gamma}_i, i \in I\}$ are initialized randomly and learned by performing a gradient descent for optimizing the following mean square rating loss:

$$\mathcal{L}_{rating}(\Theta) = \sum_{u \in U} \sum_{(i,r) \in \theta_u} [\mu + \mu_u + \mu_i + \langle \gamma_u, \gamma_i \rangle - r]^2 + \lambda \Omega(\bar{\gamma}_u, \bar{\gamma}_i) \quad (3)$$

We added the regularization term $\lambda \Omega(\bar{\gamma}_u, \bar{\gamma}_i)$ in order to avoid overfitting [16].

Given a user u and an item i, the rating score is computed using the classical matrix factorization formula proposed in [8]. Let μ, μ_u and μ_i denote respectively the overall bias, the user bias and the item bias:

$$score(u,i) = \mu + \mu_u + \mu_i + \langle \gamma_u, \gamma_i \rangle \quad (4)$$

4. EXPERIMENTS

In this section, we evaluate our model performance regarding two different tasks, item ranking and rating prediction.

Baselines

In order to evaluate our rating prediction, we implemented a matrix factorization (MF) as presented in [10] and the model presented by *McAuley et al.* (EXP) in [12] as baselines. To evaluate our model in an item ranking paradigm we used the popularity function that always return the most popular result (POP) as a prediction.

Datasets

In order to perform our experiments, we used five time-labeled datasets: *BeerAdvocate* and *RateBeer* datasets from [12] contain reviews on beers, MovieLens10m and Flixster datasets contain reviews on movies and FineFoods from [12] contains reviews from the fine foods category from Amazon. The characteristics of each dataset can be found in Table 1.

As the ratings were on different scales, we normalized them all to be on the scale $[0,5]$. Then, we used the settings presented in [19]: only the users with more than 20 reviews were kept, and every dataset has been divided in time windows, each representing a year period (except for the Flixster dataset which is divided bimonthly regarding its smaller timespan).

In order to perform the evaluation, the last three time windows were selected as test $\mathcal{U}$ and for each $\mathcal{U}$, we used all the previous windows for training $\mathcal{T}$. We ran each experiment 5 times on each window and report the average results for reliability.

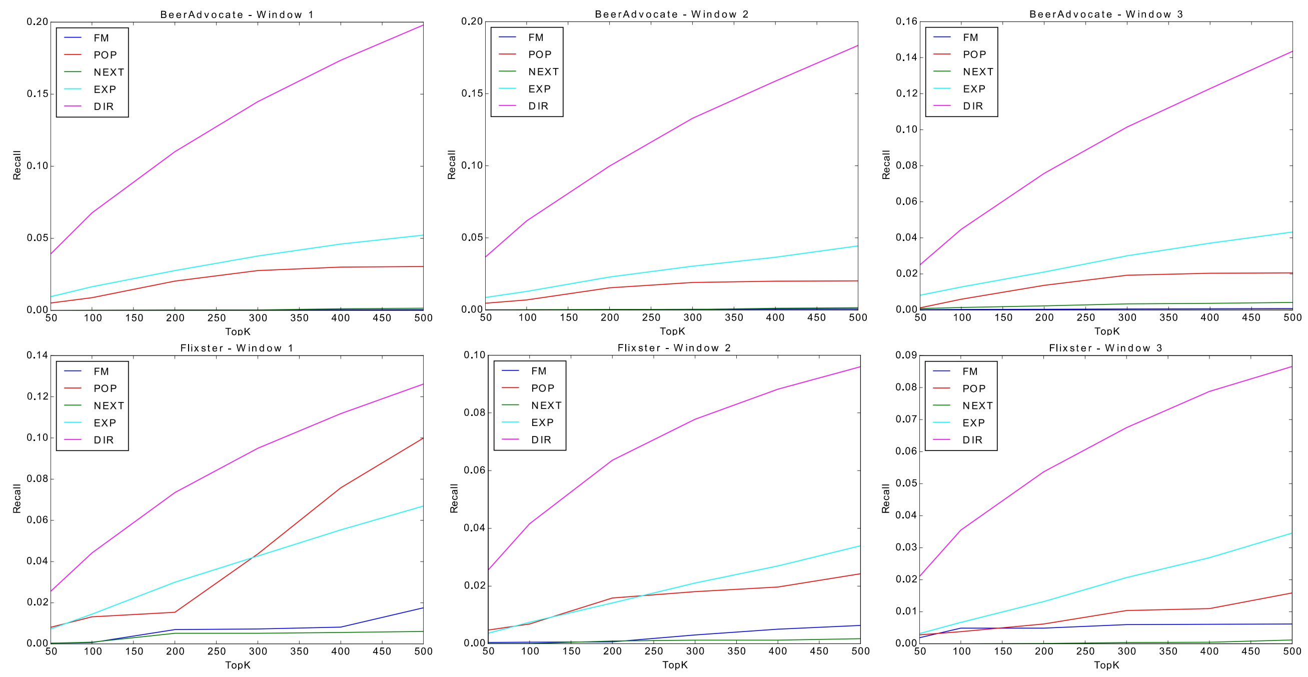

Figure 3: Results regarding the item prediction for each inference model given for K = 50, 100, 200, 300, 400 and 500 and expressed in recall.

Dataset	# items	# users	# reviews
BeerAdvocate	66051	33387	1586259
RateBeer	110419	40213	2924127
MovieLens	10000	72000	10000000
Flixster	49000	1000000	8200000
FineFoods	74258	256059	568454

Table 1: Datasets characteristics.

Evaluation

Item ranking

We evaluated our model by computing the Recall@K measure for K=300 and compared them to the baselines. The results are presented in Table 2. A plot of the evolution of recall@K for K varying between 50 and 500 on the BeerAdvocate and Flixster datasets is also available in Fig.3.

Dataset	MF	POP	EXP	DIR
BeerAdvocate	0.001	0.022	0.033	**0.126**
RateBeer	0.001	0.008	0.021	**0.106**
MovieLens	0.118	0.027	0.12	**0.15**
Flixster	0.005	0.024	0.028	**0.08**
FineFoods	0.042	0.027	0.054	**0.156**

Table 2: Ranking Results averaged over all three time windows, given in Recall@300 for the matrix factorization (MF), the experience based model (EXP), the popularity model (POP) and the directional vector based model (DIR)

Our model (DIR) clearly outperforms all other models. This corroborates the relevance of taking into account the sequential structure of the user-item interaction as a core feature. Furthermore, the MF and EXP baselines show that the rating prediction is not a good indicator for item prediction ; this validates the idea that using the same representations to compute both rating and ranking values requires some sort of tradeoff. The DIR model uses two different yet intertwined representations, freeing it from the aforementioned tradeoff constraint.

Rating prediction

The results of our experiments expressed in MSE can be found in Table 3. We compared our items to a classic Matrix factorization as described in [10] and to the EXP model.

Dataset	MF	EXP	DIR
BeerAdvocate	0.4	0.366	**0.361**
RateBeer	0.343	0.307	**0.292**
MovieLens	0.691	0.684	**0.661**
Flixster	0.892	0.892	**0.817**
FineFoods	1.365	1.337	**1.06**

Table 3: Rating Results averaged over all three time windows, given in MSE for the matrix factorization (MF), the experience based model (EXP) and the directional vector based model (DIR)

Here also, the proposed model significantly outperforms both models, showing that incorporating the order information allows learning better user and item representations.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a light and scalable latent method for recommendation using item sequence information which has been shown to be also efficient as we performed evaluations regarding two different tasks: rating prediction and item ranking.

Regarding future work, there are many paths to explore. First, we would like to find better ways to model the user representation *e.g.* using a more sophisticated prediction functions. We would also like to add an expertise notion in order to model more precisely the temporal dimension. Last, we think that touristic data follow the same kind of temporal evolution as web data and we plan to adapt this method to visit recommendation.

6. ACKNOWLEDGEMENTS

The authors would like to thank the AMMICO project (F1302017 Q - FUI AAP 13) for funding our research.

7. REFERENCES

[1] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *In Workshop on context-aware recommender systems (CARS09*, 2009.

[2] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup Workshop 2007*, pages 3–6, 2007.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[4] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, 2014.

[5] P. G. Campos, F. Diez, and A. Bellogin. Temporal rating habits: A valuable tool for rating discrimination. In *Proceedings of the 2Nd Challenge on Context-Aware Movie Recommendation*, CAMRa '11, pages 29–35, New York, NY, USA, 2011. ACM.

[6] Y. Ding and X. Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM.

[7] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

[8] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *ACM SIGKDD*, pages 426–434, 2008.

[9] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, New York, NY, USA, 2009. ACM.

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

[11] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *ACM Conference on Recommender Systems*, pages 165–172, 2013.

[12] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *World Wide Web*, 2013.

[13] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *ACM SIGIR*, pages 329–336, 2004.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[15] M. Poussevin, V. Guigue, and P. Gallinari. Extended recommendation framework: Generating the text of a user review as a personalized summary. *CoRR*, abs/1412.5448, 2014.

[16] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Cooperative Work*, 1994.

[18] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 723–732, New York, NY, USA, 2010. ACM.

[19] C. Zhang, K. Wang, H. Yu, J. Sun, and E. Lim. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 452–460, 2014.