When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

ABSTRACT

Deep learning methods have led to substantial progress in various application fields of AI, and in recent years a number of proposals were made to improve recommender systems with artificial neural networks. For the problem of making session-based recommendations, i.e., for recommending the next item in an anonymous session, Hidasi et al. recently investigated the application of recurrent neural networks with Gated Recurrent Units (GRU4REC). Assessing the true effectiveness of such novel approaches based only on what is reported in the literature is however difficult when no standard evaluation protocols are applied and when the strength of the baselines used in the performance comparison is not clear. In this work we show based on a comprehensive empirical evaluation that a heuristics-based nearest neighbor (kNN) scheme for sessions outperforms GRU4REC in the large majority of the tested configurations and datasets. Neighborhood sampling and efficient in-memory data structures ensure the scalability of the kNN method. The best results in the end were often achieved when we combine the kNN approach with GRU4REC, which shows that RNNs can leverage sequential signals in the data that cannot be detected by the co-occurrence-based kNN method.

CCS CONCEPTS

•Information systems →Recommender systems;
•General and reference →Evaluation;

KEYWORDS

Session-Based Recommendation; Deep Learning; Nearest-Neighbors

ACM Reference format:
Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *Proceedings of RecSys'17, August 27–31, 2017, Como, Italy, ,* 5 pages.
DOI: http://dx.doi.org/10.1145/3109859.3109872

1 INTRODUCTION

Deep learning approaches based on artificial neural networks have recently led to significant advances in different application fields of AI like object classification in images, speech recognition, or game playing. Their success in these fields has inspired researchers to explore the potential of adapting deep learning methods for recommendation-related problems. While there are limited works yet that show the advantages of *directly* applying deep learning methods for the rating prediction task [29, 35], artificial neural networks were, for example, used to vectorize content features from audio, video or textual item data [1, 5, 6, 10] and in particular for the problem of *session-based recommendation* [14, 15, 30, 38].

The algorithmic task in the latter scenario is to predict the next action of a user given the sequence of the actions in the current session. The problem setting, while not largely explored in the research literature, is highly relevant in practical settings. Session-based approaches are usually applied when the visitors of the site are anonymous (not logged-in) and no past interactions of the users are known. Furthermore, considering the last few user actions is also important for applications where already known users often revisit the site with a specific short-term intent [18].

In their recent work, Hidasi et al. investigated the use of recurrent neural networks (RNN) for session-based next-item recommendation [14]. RNNs are a natural choice for this problem and have been successfully explored for other sequence-based prediction problems in the past [4, 8, 9, 16]. Technically, the approach in [14] uses a customized RNN with Gated Recurrent Units (GRU). An experimental evaluation on two datasets indicated that their GRU4REC method significantly outperforms other methods, including an item-based k-nearest-neighbor (kNN) method, which was the strongest baseline in their experiments. Despite these positive results, some questions regarding the effectiveness of the GRU4REC method remain open. The experiments in [14] were, for example, only conducted on two specific datasets, used baselines whose strength cannot be easily judged, and were based on a proprietary evaluation protocol without cross-validation.

To better understand the true effectiveness of the proposed approach, we conducted a series of experiments on multiple datasets in which we benchmarked the GRU4REC method with an alternative session-based nearest neighbor method, which was identified as a strong baseline in previous works on session-based music and e-commerce recommendation problems [2, 12, 21]. To achieve higher accuracy and to scale for larger datasets, our session-based kNN method incorporates heuristics to sample suitable neighbors. Our results show that the proposed kNN method leads to the same accuracy results as the best configuration reported in [14] and outperforms GRU4REC in many other tested problem setups.[1] Combining GRU4REC with the kNN methods in a weighted hybrid approach finally often led to the best results, which indicates that RNN methods are indeed capable of capturing sequential patterns in the data that the kNN approach could not identify.

RecSys'17, August 27–31, 2017, Como, Italy

978-1-4503-4652-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3109859.3109872

[1] Recent works suggest that advanced nearest-neighbor models can lead to competitive performance also for common item-ranking tasks [34].

2 EXPERIMENT CONFIGURATIONS

2.1 Algorithms

2.1.1 GRU4REC. We used the GRU4REC implementation in Python that the authors of [14] share online.[2] The code is regularly updated by the authors and includes the implementation of the GRU4REC method, the code of their baseline algorithms, as well as the code for the evaluation procedure used in [14].

2.1.2 Session-based kNN. The kNN method takes the set of user actions in the current session, e.g., two view events for certain items, and then in a first step determines the k most similar past sessions in the training data. Then, given the current session s, the set of k nearest neighbors N_s, and a function $sim(s1, s2)$ that returns a similarity score for two sessions $s1$ and $s2$, the score of a recommendable item i is

$$score_{\text{KNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \times 1_n(i) \quad (1)$$

where $1_n(i) = 1$ if n contains i and 0 otherwise, see also [2]. We tested different distance measures. The best results were achieved when comparing the sessions, which were encoded as binary vectors of the item space, using cosine similarity.

Determining the similarity of the current session with millions of past sessions after each user action cannot easily be accomplished under the time constraints of online recommendation. We therefore pre-process the training sessions and create an in-memory index data structure (cache) on startup. Specifically, for each item we create an index that points to the sessions in which the item appears. For each session, we furthermore have a pointer to its set of items. When recommendations for a session s are needed, we first determine the set of *possible* neighbors by creating the union of sessions in which the items of s are contained. This is a fast operation as it only involves a cache lookup and set operations. From this set of possible neighbors, we create a subsample of m sessions randomly or using a heuristic. In some experiments, we took the most recent sessions in case such information was available as focusing on recent trends has shown to be effective for recommendations in e-commerce [19]. From m we select the k nearest neighbors regarding the current session s. Again through lookup and set union operations, we create the set of recommendable items R that appear in one of the k sessions. We then compute the score for the items in R using Equation 1. The set operations, similarity computations, and the final predictions can be done very efficiently, as will be discussed later in Section 3.2.3.

2.1.3 Hybrid Approach. We tested switching, cascading, as well as weighted hybrids of the GRU4REC and the kNN method. A weighted combination led to the best results in our experiments, where in the most successful configurations the kNN score was assigned a slightly higher weight. All source code and the public datasets used in our experiments can be found online. [3]

2.2 Datasets and Evaluation Protocols

We performed experiments both on variants of the ACM RecSys 2015 Challenge dataset (*RSC15* and *RSCW*) as used in [14], on the public e-commerce dataset used in the TMall competition (*TMALL*), as well as on three different datasets containing music playlists from the platforms last.fm (*LFM*), artofthemix.org (*AOTM*), and 8tracks.com (*8T*). Music playlists are different in nature from e-commerce user logs in various ways. Nonetheless, they are designed to be consumed in a listening session and the tracks are often arranged in a specific sequence by their creators. With the experiments on these datasets our goal is to assess if recurrent neural networks can capture sequential patterns in the data which are not leveraged by the co-occurrence-based kNN approach. The basic dataset statistics are shown in Table 1, where *RSCW* and *TMALL* correspond to the average characteristics when applying a sliding-window protocol, as will be described below. In [14], Hidasi et al. use GRU4REC to predict the next item view events in a session. They incrementally add events to the sessions in the test set and report the average hit rate (*HR*) and the Mean Reciprocal Rank (*MRR*) at list length 20. They trained the GRU4REC on six months of data and used one single day for the evaluation.

Table 1: Dataset characteristics

	RSC15	RSCW	TMALL	LFM	AOTM	8T
Sessions	8M	4M	650K	120K	82K	520K
Avg. length	3.97	3.92	7.5	28.24	11.48	9.20
Items	37K	34K	300K	200K	54K	200K

In addition to this procedure, we performed experiments where we created multiple train-test splits, consisting of 3 and 1 month of training data for the *RSC15* and TMall dataset, respectively, and the subsequent day as the test data.[4] This sliding-windows approach allows us to minimize the risk that the obtained results are specific to the single train-test split used in [14]. Furthermore, we used the algorithms to predict purchase events in the sessions, which was one of the original tasks of the ACM RecSys 2015 challenge. To investigate the performance of the methods at different stages of a session, we also measured the accuracy when predicting the *second* and the *last* view of each session. Finally, we varied the amount of the training data to see how the algorithms compare in case of more sparse data situations.

3 RESULTS

3.1 Accuracy Results

3.1.1 ACM RecSys 2015 Challenge Dataset. Table 2 shows the results when using the experimental configuration used in [14], ordered by the values for HR@20 as done in the original paper. We could reproduce their best hit rate and MRR results (using their optimal parameters) for the methods GRU4REC(1000,BPR) and GRU4REC(1000,TOP1), which use 1000 hidden units and the TOP1 and BPR's pairwise ranking loss function, respectively. In Table 2, we additionally include the results for list length ten, which might be more important in different application domains.

The method KNN_{MR}(500,1000), which uses the 500 nearest neighbors from the 1000 most recent candidate sessions, outperforms GRU4REC in all measurements except for the MRR in the TOP1

[2] https://github.com/hidasib/GRU4Rec
[3] http://bit.ly/2nfNldD
[4] Using larger training splits for the TMall dataset, due to its large number of items, led to prohibitively high computational costs by the GRU4REC method, which is why we report the results when using one month as training data for this dataset.

Table 2: Results when using the evaluation scheme of [14].

Method	HR@10	MRR@10	HR@20	MRR@20
WH(KNN,GRU,0.6,0.4)	**0.568**	0.256	**0.691**	0.265
WH(KNN,GRU,0.1,0.9)	**0.568**	**0.269**	0.666	**0.276**
KNN_{MR}(500,1000)	0.521	0.242	0.641	0.250
GRU4REC(1000,BPR)	0.517	0.235	0.636	0.243
GRU4REC(1000,TOP1)	0.517	0.261	0.623	0.268
KNN_{RAND}(500,1000)	0.499	0.235	0.616	0.242
GRU4REC(100,TOP1)	0.481	0.221	0.595	0.230

configuration. Combining the kNN-method with GRU4REC in a weighted approach (WH) leads to the best results. As in [14], the "winner" at list length 20 depends on the metric. Also, different weights for the hybrid method have to be applied to achieve the best results for a given list length.[5] The increases of the best configuration at length 20 are about 8% for the hit rate. A smaller increase was observed for the MRR when compared to the results of [14]. Even with a random sampling of candidate sessions (KNN_{RAND}(500,1000)) the kNN-method does not fall far behind and is consistently better than GRU4REC with 100 hidden units.

Table 3 shows the best results obtained with KNN_{MR}(500,1000) and GRU4REC(1000,TOP1) when using alternative measurements, including the sliding window protocol (*RSCW*), the prediction of the second (*SECOND*) and last item view (*LAST*), and the accuracy when predicting which item is purchased in a session (*BUYS*). In all experiments, the kNN method outperforms GRU4REC. For the sliding windows protocol, we applied the Wilcoxon signed-rank test over the five experiment runs, which revealed that the differences are statistically significant in terms of the hit rate ($\alpha = 0.05$).

3.1.2 Results for Other Datasets. Table 4 shows the results for the additional datasets. We again report the results at list length 20 for GRU4REC(1000,TOP1) and KNN(500,1000). For the *TMALL* dataset, we applied the sliding-window approach as for *RSCW*. Note that in this dataset each session is defined as the sequence of actions of a user during one day (resulting in a larger average session length as shown in Table 1). As the playlists have no timestamp attached, we randomly assigned each playlist to one of 31 buckets (days of the month) and used one of them as test data.

In the majority of the configurations, the kNN method outperformed GRU4REC both in terms of the hit rate and the MRR, in many cases strongly. Only on the last.fm dataset, GRU4REC worked better than the kNN approach; on the AOTM dataset, GRU4REC was furthermore better in terms of the MRR. Generally, however, the results obtained for the additional datasets indicate that the good performance of the kNN method on the *RSC15* dataset is not due to the specific nature of this dataset or the application domain.

3.2 Additional Analyses

3.2.1 Focusing on Recent Data. The *RSC15* dataset contains about 45,000 sessions per day. Since our kNN method only uses, e.g., one thousand *possible* neighbor sessions, which are not necessarily all from the same day, we were interested to what extent we can omit past sessions without compromising the accuracy results.

[5] In all experiments we tuned the parameters for the different algorithms using grid search. We optimized the hit rate on validation sets (subsets of the training sets). Due to the run time GRU4REC was only optimized with 100 layers as also done in [14].

Table 3: Additional measurements for the RSC15 dataset.

Dataset	RSCW		SECOND		LAST		BUYS	
M@20	HR	MRR	HR	MRR	HR	MRR	HR	MRR
KNN	0.621	0.267	0.716	0.355	0.446	0.196	0.758	0.290
GRU4REC	0.587	0.261	0.655	0.300	0.388	0.174	0.542	0.215

Table 4: Results for other datasets.

Dataset	TMALL		LFM		AOTM		8T	
M@20	HR	MRR	HR	MRR	HR	MRR	HR	MRR
KNN	0.370	0.171	0.078	0.011	0.068	0.008	0.050	0.010
GRU4REC	0.223	0.117	0.121	0.053	0.035	0.012	0.019	0.008

Figure 1 shows MRR results when we incrementally remove data from the training set, beginning with the oldest sessions. The values for the hit rate are similar. For the GRU4REC method, it seems sufficient to focus on the last 60 days. For the kNN method in contrast, it is sufficient to retain the information of the last two days. While this might come surprising, focusing on the most recent events has shown to be effective in the past in the domains of e-commerce and news recommendations [19, 23].

As mentioned above, we made additional experiments on the *RSC15* data and repeatedly sampled 1000 random neighbor sessions instead of the most recent ones. The average results were shown in the last row of Table 2. Using random neighborhood sampling leads to slightly lower accuracy results, which are however still higher than the ones obtained by GRU4REC with 100 hidden units.

3.2.2 Popularity Bias and Catalog Coverage. To assess possible recommendation biases, we measured the average popularity of the recommendations. Using a normalized popularity score for the top-20 recommendations (in the setup of [14]) we found that the kNN method tends to recommends slightly more popular items on average than GRU4REC (0.036 vs. 0.028). With respect to catalog coverage, we observed that the top-20 recommendations of GRU4REC include about 47% of the items at least once, which is slightly more than the kNN method covers (41%). This latter observation is not surprising, given the focus of the kNN method on

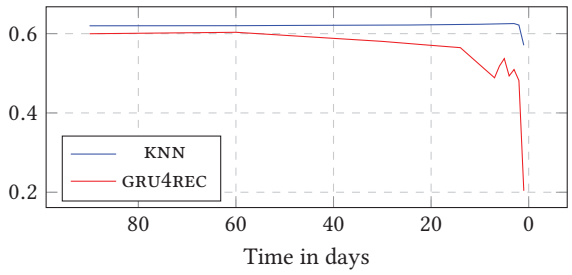

Figure 1: MRR@20 for the RSC15 when artificially reducing the size of the training set from 3 month to 1 day.

the last few days. Alternative neighbor sampling strategies can however be designed to deal, e.g., with such accuracy-coverage trade-offs.

3.2.3 Computational Complexity and Memory Usage. On a desktop computer with an Intel i7-4790k processor, training GRU4REC in the best configuration needed about 23 hours for the *RSC15* dataset, which can be reduced to about 8 hours when calculations are performed by the GPU (Nvidia GeForce GTX 960). The kNN method needs about 90 seconds to build the in-memory data structures. Creating one recommendation list with GRU4REC needed about 12 ms on average and 26 ms for the kNN method. Overall, computing all (about 40,000) recommendations needs about 27 minutes with the kNN method including data structure initialization, whereas GRU4REC needs more than 8 hours in total. At the same time, the kNN method has the advantage of supporting online updates. Further speed-ups for the kNN approach can in theory be achieved when the similarity computations are parallelized, since these calculations can be done independently for individual subsets of the neighbor candidates.

The data structures of the kNN method occupy about 6.4 GB of main memory when the entire *RSC15* training dataset (2.3 GB of raw data) is used. No special data structures are used in our implementation and given the observations from above, keeping only a specific amount of the most recent log actions will help to reduce the memory requirements. For the same dataset, GRU4REC's model needs about 60 MB for 100 and 600 MB for 1000 hidden units. GRU4REC's memory demand is thus dependent of the algorithm parameters and significantly increases with the number of items. In the playlist and TMall experiments, GRU4REC's memory demands exceeded the capacity of our graphic card, making computations very slow, which is why we could only make a limited number of evaluation runs on these datasets.

4 RELATED WORK

The commonly used algorithmic approaches to leverage sequential information for predicting next user actions include Markov models and sequential pattern mining techniques. In one of the earlier works on this topic, Shani et al., for example, proposed the application of Markov Decision Processes in the scenario of an online book store [27]. More recent works that rely in Markov models include [3, 13, 17, 26] or [31]. In [31], Markov models were, for example, used with the goal to detect topic sequences in user sessions for the next-item prediction task. Detecting and leveraging sequences of topics was also the goal in [11] for the next-track music recommendation scenario. In this case, however, the authors applied sequential pattern mining techniques. Sequential patterns have been investigated earlier also for the problem of predicting the online navigation behavior of users, e.g., in [25].

Approaches based on Markov models and sequential pattern mining represent alternative baselines for session-based recommendations. Not all approaches based on Markov model however scale too well for large datasets [27]. Sequential pattern mining approaches (and association rule mining approaches in general) often require some effort to find suitable thresholds for rule learning and, depending on the application domain, do not lead to better accuracy results than co-occurrence-based kNN methods [2].

Particularly in recent years, RNNs have been applied in different ways to leverage sequence information when recommending, often based on ideas of the LSTM model proposed in [16] to avoid the vanishing or exploding gradient problem.

Zhang et al. [38] for example successfully applied RNNs in a related setting of predicting sequential advertisement clicks, in which the partitioning into sessions is less important. Hidasi et al., who proposed GRU4REC [14], were among the first to explore RNNs for session-based recommendations. Using the same evaluation protocol and *RSC15* data set, Tan et al. in [30] proposed an enhanced version of GRU4REC, which is trained on embedded item sequences with random drop-out to reduce overfitting. The models are also retrained on the most recent sessions to better adapt to short-term trends. In their paper, the authors report significant improvements over [14] which would also slightly outperform our hybrid approach. The work confirms that considering recent trends is helpful for this *RSC15* dataset and domain. Experiments on other datasets were unfortunately not reported. Since no source code was available, we re-implemented their method based on the information in the paper, but could not reproduce their results.

In [15] and [32] RNN-based approaches were recently proposed that leverage additional item features to achieve higher accuracy. Including additional information about users and items in the recommendation process is also possible for kNN based methods, but was not in the focus of our work.

Combining short-term with long-term models has shown to be a successful strategy in the past, e.g., in the e-commerce domain [18]. Song et al. [28] recently proposed an approach to predicting news click sequences, which used an RNN to model short-term interests in a combination with long-term models. The kNN method used in this paper can be combined with long-term models as well. However, in the case of the *RSC15* dataset, where sessions have no user ID attached, no long-term user models can be learned.

Yu et al. [37] use RNNs for the related task of next basket recommendations, in which, e.g., items for a shopping cart are suggested based on a user's history of past shopping carts. Although the scenario is slightly different from the one in this paper, the kNN method can be easily adapted as a baseline for this scenario.

In general, NNs have been used for a number of recommendation-related tasks in recent years. Often, such networks are used to learn embeddings of content features in compact fixed-size latent vectors, e.g., for music, for images, for video data, for documents, or to represent the user [1, 5–7, 10, 20, 24, 36]. These representations are then integrated, e.g., in content-based approaches, in variations of latent factor models, or are part of new methods for computing recommendations [6, 7, 10, 22, 29, 33, 35].

5 CONCLUSIONS

Our work shows that nearest-neighbor methods should be considered as competitive baselines for session-based recommendation scenarios. Considering a mix of co-occurrence signals and sequential patterns, as identified by recurrent neural networks, led to the best results in our experiments. In practice, however, one has to decide based on the application domain if the obtained accuracy gains justify the usage of more complex learning methods. Our future works include the comparison of the performance of additional session-based algorithms like [26] or [30].

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 107–114. DOI: http://dx.doi.org/10.1145/2959100.2959180

[2] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *Computing Surveys* 47, 2 (Nov. 2014), 26:1–26:35. DOI: http://dx.doi.org/10.1145/2652481

[3] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, 714–722. DOI: http://dx.doi.org/10.1145/2339530.2339643

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). http://arxiv.org/abs/1412.3555

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 191–198. DOI: http://dx.doi.org/10.1145/2959100.2959190

[6] Sander Dieleman. 2016. Deep learning for audio-based music recommendation. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 1–1. DOI: http://dx.doi.org/10.1145/2988450.2991128

[7] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. ACM, 278–288. DOI: http://dx.doi.org/10.1145/2736277.2741667

[8] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179 – 211. DOI: http://dx.doi.org/10.1016/0364-0213(90)90002-E

[9] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850

[10] Yupeng Gu, Bo Zhao, David Hardtke, and Yizhou Sun. 2016. Learning Global Term Weights for Content-based Recommender Systems. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. ACM, 391–400. DOI: http://dx.doi.org/10.1145/2872427.2883069

[11] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. ACM, 131–138. DOI: http://dx.doi.org/10.1145/2365952.2365979

[12] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting to User Preference Changes in Interactive Recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI '15)*. AAAI, 4268–4274. http://dl.acm.org/citation.cfm?id=2832747.2832852

[13] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE '09)*. IEEE, 1443–1454. DOI: http://dx.doi.org/10.1109/ICDE.2009.71

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR '16)*. ACM. http://arxiv.org/abs/1511.06939

[15] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 241–248. DOI: http://dx.doi.org/10.1145/2959100.2959167

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. DOI: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[17] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, 241–244. DOI: http://dx.doi.org/10.1145/2792838.2799684

[18] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, 211–218. DOI: http://dx.doi.org/10.1145/2792838.2800176

[19] Dietmar Jannach and Malte Ludewig. 2017. Determining Characteristics of Successful Recommendations from Log Data – A Case Study. In *Proceedings of the 32th Annual ACM Symposium on Applied Computing (SAC '17)*. ACM.

[20] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 233–240. DOI: http://dx.doi.org/10.1145/2959100.2959165

[21] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders Within E-Commerce Recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP '16)*. ACM, 27–35. DOI: http://dx.doi.org/10.1145/2930238.2930244

[22] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, 811–820. DOI: http://dx.doi.org/10.1145/2806416.2806527

[23] Cornelius A. Ludmann and H.-Jürgen Appelrath. 2016. Lessons Learned from Using a Data Stream Management System for Real-time Recommendation of Popular News Articles based on Real User Streams. In *Proceedings of the 4th Workshop on Large-Scale Recommender Systems at ACM RecSys 2016 (LSRS '16)*. ACM.

[24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, 43–52. DOI: http://dx.doi.org/10.1145/2766462.2767755

[25] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM '02)*. IEEE, 669–672. DOI: http://dx.doi.org/10.1109/ICDM.2002.1184025

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, 811–820. DOI: http://dx.doi.org/10.1145/1772690.1772773

[27] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *The Journal of Machine Learning Research* 6 (Dec. 2005), 1265–1295. http://dl.acm.org/citation.cfm?id=1046920.1088715

[28] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-Rate Deep Learning for Temporal Recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM Press, 909–912. DOI: http://dx.doi.org/10.1145/2911451.2914726

[29] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 11–16. DOI: http://dx.doi.org/10.1145/2988450.2988456

[30] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 17–22. DOI: http://dx.doi.org/10.1145/2988450.2988452

[31] Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for Detecting Topics of User Sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, 33–40. DOI: http://dx.doi.org/10.1145/2645710.2645739

[32] Bart lomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 273–276. DOI: http://dx.doi.org/10.1145/2959100.2959162

[33] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 225–232. DOI: http://dx.doi.org/10.1145/2959100.2959160 arXiv:1607.07326

[34] Koen Verstrepen and Bart Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2645710.2645731

[35] Jeroen B. P. Vuurens, Martha Larson, and Arjen P. de Vries. 2016. Exploring Deep Space: Learning Personalized Ranking in a Semantic Space. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 23–28. DOI: http://dx.doi.org/10.1145/2988450.2988457

[36] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, 1235–1244. DOI: http://dx.doi.org/10.1145/2783258.2783273 arXiv:1409.2944

[37] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, 729–732. DOI: http://dx.doi.org/10.1145/2911451.2914683

[38] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*. AAAI, 1369–1375. http://dl.acm.org/citation.cfm?id=2893873.2894086