Improving Session Recommendation with Recurrent Neural Networks by Exploiting Dwell Time

Alexander Dallmann
Alexander Grimm
Christian Pölitz
Daniel Zoller
dallmann@informatik.uni-wuerzburg.de
alexander.grimm@stud-mail.uni-wuerzburg.de
poelitz@informatik.uni-wuerzburg.de
zoller@informatik.uni-wuerzburg.de
University of Würzburg
Data Mining and Information Retrieval (DMIR) Group
Am Hubland, Informatik
97074, Germany, Würzburg

Andreas Hotho*
hotho@informatik.uni-wuerzburg.de
University of Würzburg
Data Mining and Information Retrieval (DMIR) Group
Am Hubland, Informatik
97074, Germany, Würzburg

ABSTRACT

Recently, Recurrent Neural Networks (RNNs) have been applied to the task of session-based recommendation. These approaches use RNNs to predict the next item in a user session based on the previously visited items. While some approaches consider additional item properties, we argue that item dwell time can be used as an implicit measure of user interest to improve session-based item recommendations. We propose an extension to existing RNN approaches that captures user dwell time in addition to the visited items and show that recommendation performance can be improved. Additionally, we investigate the usefulness of a single validation split for model selection in the case of minor improvements and find that in our case the best model is not selected and a fold-like study with different validation sets is necessary to ensure the selection of the best model.

KEYWORDS

session recommendation, neural networks, user interest modeling, dwell time, recurrent neural networks

1 INTRODUCTION

Today, sales in the Internet are increasing rapidly, but supporting customers while they're shopping is still a challenge as users are generally hidden by the anonymity of the web. One working solution are recommender systems which identify a user's needs by analyzing the shopping history and the user's behavior. Therefore, tracking the user becomes an essential tool as it allows to understand the behavior and goals of the customer and ensures the growth of the business.

Recommendation can be done using explicit information from a user's purchase history, but is then limited to the general past preferences, while the current interest, for example in a new product, is hidden. However, the current interest of a user can be gleaned from the session and traces left by the user while searching for new products in a web shop. These traces left by the user are only implicit

feedback which tends to be noisy and difficult to analyze. Recently, advanced recommender systems utilize deep learning approaches when analyzing such click traces to predict the way through the shop toward the next purchase. Typically, a user checks prices and other properties of the next product of interest and investigates similar products. Therefore, it is a plausible assumption that the next product to buy will be visited by the user while browsing in the shop and can therefore be identified by such learning methods.

Recent approaches utilize the order of visited items as good indicators for predicting the next click on a page or item. Additionally, such a session contains information about the type of products and the frequency of visits. When analyzing a user's behavior, one could observe that some products are investigated in more detail than others. This different interest is to some extent reflected by the time a user spends with the product before investigating the next one by following a link and is not only expressed by the frequency of the visits within a session. In this work, we will make use of this additional time information called dwell time [20] and show that deep learning networks can make use of this additional information to improve the recommendation quality.

We propose an extension to a state-of-the-art session-based recurrent neural network model that integrates the item *dwell time* into the model. Additionally, we show that care must be taken when conducting a hyper-parameter study to ensure the selection of the best parameters and present a fold-like scheme for selecting the model. Finally, results are shown verifying that the dwell time positively impacts recommendation performance.

The rest of the paper is structured as follows. In Section 2 we discuss related work. Next we describe the dataset and give details about the applied preprocessing in Section 3. This is followed by a description of the studied models in Section 4. Results of our experiments are shown in Section 5 followed by a discussion in Section 6. Finally we give a conclusion in Section 7.

2 RELATED WORK

Recommendation systems help by suggesting resources (items) in (web) applications based on user and resource preferences [1]. Session recommendation systems learn a model of a user's behavior by

---

*Also with L3S Research Center.

using a session of events (requested resources, products, pages or more generally items) that are generated by the user. For example, Item-based recommendations, as in [11], use similar item profiles to recommend new items in the current session. Session recommendation can also be treated as a sequence learning problem that can be modeled by a Markov Decision Process which predicts possible next events (items) in a given session [15] and hence recommend corresponding resources.

These approaches are in contrast to collaborative filtering [2] approaches that factorize user information to recommend based on similar user profiles or similarity-based approaches that cluster users [7] or items [14] based on their profiles to extract preferences.

Recently, (deep) neural networks have been used for recommendation systems [13]. These networks learn feature representations of items, users or whole sessions. [19], for example, learn latent feature representations of content information with stacked denoising autoencoders, [12] apply convolutional neural networks on audio content for music recommendation, [6] use item embeddings and [4] use embeddings of videos, search query terms and user features on Youtube. Different combinations of network architectures have been proposed in the literature. For instance, [3] investigate combining deep (neural network) or wide (linear models) architectures to capture generalization capabilities of a deep neural network together with modeling strength of sparse feature interaction of linear models. [18] use a feed forward neural network to encode item information and an RNN to encode session information for session-based recommendations. [16] explicitly model temporal behavior by RNNs together with user and item features using feed forward nets to perform temporal recommendations.

The sequential nature of most of the systems for session-based recommendations makes recurrent networks a good model candidate. Recurrent Neural Networks (RNNs) can be used to make predictions on sequential input data. Recently, Hidasi et al. proposed a sequence-to-sequence RNN model for predicting the user session from the sequence of clicks [8]. In contrast, Tan et al. use an RNN model that predicted only the last click in a session and studied the impact of item embeddings and data augmentation on the prediction performance [17]. Considering a large set of additional features to the items, [9] propose parallel RNNs. Further, temporal features like the date or time of the current session have been used in [5] to recommend music. In contrast to this approach, we use the dwell time on a per item basis as a dwell time profile for the session. Dwell time as a user feature for personalized recommendations is investigated in [20]. The dwell time is used to express preferences for a certain time. Hence, the longer the dwell time, the more relevant the item is to a user. Here, the dwell times are only used to rank the items for recommendations.

3 DATASET AND PREPROCESSING

We evaluate all models on the open RecSys Challenge 2015 dataset. The dataset contains click events from an e-commerce store that can be aggregated to user sessions and was published as part of the RecSys Challenge 2015[1]. Overall, the dataset contains 9 249 729 sessions with 33 003 944 clicks and 52 739 unique items. Table 1 shows the different attributes that are collected for each click.

Table 1: Relevant properties of the RecSys15 dataset.

Property	Description
sid	an id for the session this click event belongs to
timestamp	the time the click event was recorded
item	the id for the item that was clicked
category	the item category

In general, we apply the same preprocessing steps as described in [8]. More specifically, the dataset is split into a train and test set and the test set contains all sessions of the last day. During the analysis of the evaluation in the parameter study, with split the dataset as follows. For a fold-like validation scheme, we create six splits of the training set with each split using one of the last six days as the validation set. Additionally, all sequences with length $l < 2$ and items with a support $sup < 5$ are removed. We also remove items from the validation and test sets that don't occur in the respective training sets. As proposed in [17], the session length (l) is restricted and chosen so that $max(l) = 16$ and all sessions for which $l > 16$ are removed. This captures approximately 98% of all sessions. The properties of the train and test sets after preprocessing are listed in Table 2. We also produce a second dataset with augmented sessions as described in [17], where for each session every prefix is added as a separate example.

Table 2: Number of session, items and the avg. session length of the training and test sets.

	Train	Test
#sessions	12,864,743	30,484
#items	53,308,101	136,150
session_length	4.14	4.47

4 METHODS

In this section, we first give a description of the model in Section 4.1.

Then we shortly describe the previously proposed approaches for session-based recommendation using RNNs in Section 4.2. Next, we present our extension for integrating dwell time information in Section 4.3.

4.1 Problem Setting

We consider the task of predicting the next item in a session based on its previous items in the same session. Let a session S be an ordered series of clicks $S = (c_1, c_2, ..., c_k)$ with length k and a click c be a tuple $c = (i, t)$, where $i \in I$ is a unique identifier for the clicked item and t contains a timestamp when this click occurred. Our task is then to fit a Model M defined as $\mathbf{y} = M(\mathbf{x})$ that computes a probability distribution $\mathbf{y} \in \mathbf{R}^{|I|}$ from an input sequence $\mathbf{x} = (c_1, c_2, ..., c_k)$.

4.2 Recurrent Neural Networks Based on Item Sequences

Recently, different RNN architectures have been successfully proposed for this task. In [8], a sequence-to-sequence RNN with Gated Recurrent Unit (GRU) layers is proposed that uses a session-parallel scheme to process the input sessions. Another model that, in contrast, predicts only the next item i_k in a sequence $[i_1, i_2, \cdots, i_{k-1}]$ is studied in [17]. In order to facilitate the prediction of every item in a session, a data augmentation scheme is introduced where every session prefix is used as a distinct sample. Additionally, item embeddings are used as inputs and improvements over [8] in terms of Recall@20 and MRR@20 are reported.

Because of the better adaptability to our task and the reported results we use a model similar to the one described in [17] for solely item-based recommendation and call it IT-RNN. Specifically, we use item embeddings as input to an RNN consisting of GRU layers and predict the next item i_k in a fixed sequence of length $l = k - 1$. The model produces a probability distribution over all items using a single fully-connected layer followed by softmax as the final layer. In preliminary experiments, it turns out that this setting is superior to all the other tested configurations. An illustration of the IT-RNN model its shown in Figure 1.
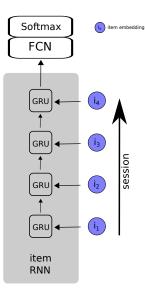

Figure 1: An illustration of the IT-RNN model for predicting the next item in a sequence given only the items as input.

4.3 Recurrent Neural Network Combining Item and Dwell Time Sequences

The described model predicts the next item in the sequences only based on previously visited items. However, the amount of time a user spends with an item (dwell time) can be an important metric for user engagement and interest [20].

For every session, we compute the dwell time for item i_k in a session as $d_k = t_{k+1} - t_k$ and get an aligned sequence of dwell times that characterizes the user interest over the items.

While dwell times are computed from timestamps and are therefore measured in milliseconds, such a high resolution is unlikely to be interesting when measuring user interest.
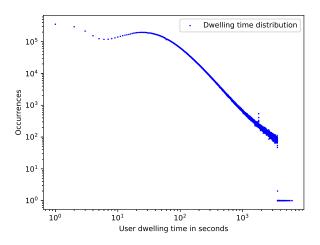

Figure 2: Distribution of dwell times in the preprocessed dataset.

Therefore, we propose a simple bucketing technique, where dwell times are rounded to the nearest second. The resulting distribution of the rounded dwell times for the RecSys Challenge 2015 dataset is shown in Figure 2. It already shows some interesting artifacts, like the two peaks. One with a very short dwell time and one at approximately 35s. After that, we observe an expected decrease in the log-log plot which only contains some minor exceptions. Using each second as a label, the dwell time for each item in a session can be encoded as a discrete class.

In most settings the captured dwell time will have some upper limit that is enforced by the consumed service, e.g. a connection timeout. However, since the upper bound is arbitrary, the representation can become overly sparse and computations could become inefficient. Furthermore, we assume that e.g. the peak with a very short time somehow encodes skipping the visited pages. The encoding of such an information is important, but rather difficult. In order to encode the hidden information of the distribution and to avoid problems with dwell times of arbitrary length, we encode the dwell time classes into continuous lower dimensional embeddings.

To capture a sense of user interest over a session, we use the sequence of dwell time embeddings as input to an RNN with a GRU layer. The output of the RNN at each step in the sequence is concatenated with the embedded item at the same step and the result is used as input to the IT-RNN part (Please keep in mind that the concatenated vectors are of different size). This way the interest a user expresses over a session can act as a weight that boosts or dampens the influence an item has on the recommendation of the next item. We call this model DT-RNN, an illustration is shown in Figure 3.

5 EXPERIMENTS

In this section, the experiments and results for our model are presented. All experiments are conducted using the dataset described
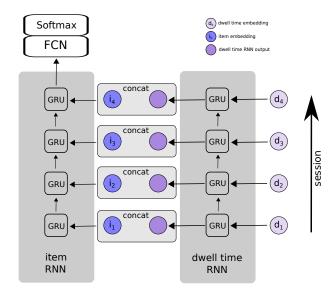
Figure 3: The DT-RNN network architecture.

in Section 3. The models are implemented in Tensorflow[2] and are trained for six epochs over the training data on a NVidia GTX 1080 Founders Edition graphics card. Batch size is set to the maximum possible value for each model depending on the model size. Adam [10] is used as the optimizer with the default settings provided by Tensorflow. For evaluation, Recall@20 and MRR@20 (Mean Reciprocal Rank) are calculated on the test set for each model and epoch, and evaluation scores of the best epoch are reported.

We want to study whether our model can use the dwell time information in addition to the item sequence to more accurately predict a user's next click and hence improve recommendation performance and user experience. Therefore, we first investigate the effects of item embeddings and augmentation on the item sequence based IT-RNN and fix the relevant hyperparameters for both models. Next, we perform a parameter study on the remaining parameters of the DT-RNN model and present the results.

5.1 Parameter Selection for IT-RNN

Results in [8] and [17] show that the best performance is achieved with an RNN layer size of 100 when comparing to a larger network with a layer of size 1000. We therefore fix the layer for IT-RNN to a similar size of 128.

Both one-hot codings and embeddings have been studied for encoding the items as input to the RNN with mixed results in [8] and [17]. Hence, we experiment with both embeddings of size 128 and one-hot codings and find that embeddings clearly outperform one-hot codings in our setup.

Next, we experiment with augmented sessions as introduced in [17] and find that training with augmented datasets improves the performance of the network on the test set. Table 3 shows the Recall@20 values with and without augmentation for both embedded and one-hot coded items.

Table 3: Comparison of one-hot codings versus embeddings for IT-RNN with $it\_rnn\_size = 128$ and $item\_em\_size = 128$. Both networks are first trained without augmenation and then with augmentation and Recall@20 is computed on the augmented test set in both cases.

	one-hot	embedding
with augmentation	0.648 4	0.687 1
without augmentation	0.629 4	0.653 3

Table 4: Results for IT-RNN with $item\_em\_size = 128$ and $it\_rnn\_size = 128$.

Method	R@20	MRR@20
IT-RNN	0.687 1	0.282 9

Based on the results we fix the size of the item embeddings to $item\_em\_size = 128$ and use the augmented dataset for training and evaluation. The evaluation results on the test set for IT-RNN are reported in Table 4 and will be used as a baseline to judge whether the proposed DT-RNN model can improve recommendation performance.

5.2 Influence of Dwell Time

We want to study the effect of dwell time on the performance compared to the IT-RNN model. Hence, we fix the parameters for the IT-RNN part and only conduct the parameter studies for the remaining parameters of the DT-RNN model. We perform a grid search for the dwell time embedding size dt_em_size and the dwell time rnn size dt_rnn_size with values for dt_em_size chosen from $[4, 8, 16, 32]$ and values for dt_rnn_size selected from $[4, 8, 16, 32, 64, 128]$.

First, we use the second to last day (sixth) day as the validation set and the remaining days for training (as given in Section 3) and run a grid search with the former specified values. Based on the evaluation on the validation set, $dt\_rnn\_size = 16$ and $dt\_em\_size = 8$ are selected as the best performing hyper-parameters and a model using these parameters is trained on the full training set and evaluated on the test set. The results are listed as *DT-RNN* in Table 5. A comparison between the results of IT-RNN in Table 4 and DT-RNN in Table 5 with the selected parameters shows no improvement over IT-RNN.

From results in [20] we know that we should not expect big improvements. However, if the expected improvements are only small, it is possible that using an arbitrary and arguably small validation set might account for the selection of the wrong model, which hides a possible gain. We therefore investigate whether a parameter setting in our grid exists that shows significant improvement over the plain IT-RNN by conducting a grid search on the full training set using the test set as the validation set. Surprisingly, the best performing setting is different from before and is listed in Table 5 as *DT-RNN**. A Wilcoxon signed rank test showed that the improvement over IT-RNN in Rec@20 is statistically significant with $p < 0.01$, while no significant improvement for MRR@20 can be observed.

Table 5: Metrics on the test set calculated for the settings selected by the parameter study $DT-RNN$ and the best settings on the test set $DT-RNN^*$.

Method	dt_em_size	dt_rnn_size	Rec@20	MRR@20
$DT-RNN$	16	8	0.687 3	0.281 0
$DT-RNN^*$	32	8	0.692 6	0.283 6

Table 6: Rec@20 values with different days chosen as the validation set. The hyper parameters selected by a parameter study $dt\_em\_size = 16$ on a fixed validation day (here day 6^*) are compared to those with the best performance on the test set $dt\_em\_size = 32$.

Day	dt_rnn_size	Rec@20	
		$dt\_em\_size = 16$	$dt\_em\_size = 32$
1	8	0.636 5	**0.639 2**
2	8	0.629 6	**0.631 9**
3	8	0.661 1	**0.665 5**
4	8	**0.673 8**	0.673 6
5	8	0.684 6	**0.688 5**
6*	8	**0.666 1**	0.663 4
$\overline{Rec@20}$		0.658 6	**0.660 4**

The different model selection results could be caused by parameter overfitting or by an arbitrary small validation set. To decide this question, we conduct a third parameter study and train both parameter settings in Table 5 on six different train-validation splits as described in Section 3. Table 6 shows the results for the six different splits. Only two of them, including our initial validation set, show a better performance with the initially chosen parameter settings ($dt\_rnn\_size = 16$), while the majority of four splits shows higher performance for the optimal setting ($dt\_rnn\_size = 32$). Calculating the average Rec@20 values for both models over all splits shows that on average the best performing model *DT-RNN** should have been selected with $dt\_rnn\_size = 32$ which results in a significant improvement of our approach.

6 DISCUSSION

Our experiments show that the proposed extension can exploit dwell times to boost the performance in terms of Rec@20. Unfortunately, the gain is rather limited and seems to be hidden by other effects, as our analysis reveals.

Selecting the correct model in a parameter study is not straightforward and is also influenced by the selected parts of the dataset used for training and validation. Conducting the study on a small and limited validation set can yield the wrong parameter combination if the improvement is small, as in our case. One solution can be to use several folds or a complete leave-one-out scheme. We experimented successfully with using different days as validation sets, but a more detailed analysis is also needed for experiments relying on the same split of the data as we used to check the generality of the results.

The used dataset split is also of interest in other settings, as a poorly chosen validation set could lead to results which are not applicable in general. This could be caused by several reasons. When dealing with temporal data, such data typically contains seasonally or weekly effects, i.e. buy patterns which show a significantly different behavior on different time slots. In this case, one can no longer assume that the distribution of the data is the same, which could be the reason for the observed results in our experiments. Therefore, care must be taken when evaluating models on small and timely restricted datasets.

7 CONCLUSION

In this work, we showed that the performance of an RNN for session-based recommendation can be improved by integrating item dwell times into the model. Additionally, we demonstrated that using a single validation set for a parameter study can lead to a sub optimal choice of parameters if the performance gain is only small. Furthermore, we showed that using a fold-like scheme with several validation sets can help to find the optimal parameters in this case.

In future work, we would like to provide a more detailed analysis of temporal effects in datasets comprised of user sessions on model selection in parameter studies and find general guidelines for performing these studies. Additionally, we want to investigate the effectiveness of our model on different datasets from various domains. Finally, we would like to evaluate the effectiveness of incorporating user dwell time as a measure of user interest in other settings, e.g. buy event prediction.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. 2013. Recommender Systems Survey. *Know.-Based Syst.* 46 (July 2013), 109–132. DOI: https://doi.org/10.1016/j.knosys.2013.03.012

[2] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 43–52. http://dl.acm.org/citation.cfm?id=2074094.2074100

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* abs/1606.07792 (2016). http://arxiv.org/abs/1606.07792

[4] 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, New York, NY, USA, 191–198. DOI: https://doi.org/10.1145/2959100.2959190

[5] Ricardo Dias and Manuel J. Fonseca. 2013. Improving Music Recommendation in Session-Based Collaborative Filtering by Using Temporal Context. In *Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI '13)*. IEEE Computer Society, Washington, DC, USA, 783–788. DOI: https://doi.org/10.1109/ICTAI.2013.120

[6] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-Based Recommendations Using Item Embedding. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces (IUI '17)*. ACM, New York, NY, USA, 629–633. DOI: https://doi.org/10.1145/3025171.3025197

[7] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*. ACM, New York, NY, USA, 230–237. DOI: https://doi.org/10.1145/312624.312682

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015).

[9] 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, New York, NY, USA, 241–248. `DOI`:https://doi.org/10.1145/2959100.2959167

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[11] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (Jan 2003), 76–80. `DOI`:https://doi.org/10.1109/MIC.2003.1167344

[12] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep Content-based Music Recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*. Curran Associates Inc., USA, 2643–2651. http://dl.acm.org/citation.cfm?id=2999792.2999907

[13] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. ACM, New York, NY, USA, 791–798. `DOI`:https://doi.org/10.1145/1273496.1273596

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295. `DOI`:https://doi.org/10.1145/371920.372071

[15] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (Dec. 2005), 1265–1295. http://dl.acm.org/citation.cfm?id=1046920.1088715

[16] 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, New York, NY, USA, 909–912. `DOI`:https://doi.org/10.1145/2911451.2914726

[17] 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 2016)*. ACM, New York, NY, USA, 17–22. `DOI`:https://doi.org/10.1145/2988450.2988452

[18] Bartlomiej 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, New York, NY, USA, 273–276. `DOI`:https://doi.org/10.1145/2959100.2959162

[19] 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, New York, NY, USA, 1235–1244. `DOI`: https://doi.org/10.1145/2783258.2783273

[20] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond Clicks: Dwell Time for Personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 113–120. `DOI`:https://doi.org/10.1145/2645710.2645724