Learning to Rank Hotels for Search and Recommendation from Session-based Interaction Logs and Meta Data

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

Dietmar Jannach
University of Klagenfurt
Austria
dietmar.jannach@aau.at

ABSTRACT

Being able to provide high quality search and recommendation services can be a decisive success factor for online applications, e.g., in today's competitive e-commerce environments. Context-adaptive and personalized item suggestions can help to both improve the user experience and the provider's short-term and long-term revenue. However, automating this form of adaptation can be challenging, when no long-term preference profiles are available. In these situations, the user's preferences and short-term intent must be derived from the last few observed interactions.

In this work, we present a hybrid approach to rank hotels based on the user's most recent interactions and meta data about the available items. The developed recommendation approach can be used both for personalized search and session-based recommendation. Technically, we employed a combination of a gradient-boosted learning-to-rank model, Bayesian Personalized Ranking and an embedding model using Doc2Vec. The approach was successfully evaluated in the context of the ACM RecSys 2019 challenge, where it led our team *letoh govatri* to the fifth place on the leaderboard, with a ranking accuracy only 0.53% below the winning approach.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**.

KEYWORDS

Search Personalization; Recommendation; Cold Start; Session-based Recommendation; Hotels; Travel and Tourism

ACM Reference Format:
Malte Ludewig and Dietmar Jannach. 2019. Learning to Rank Hotels for Search and Recommendation from Session-based Interaction Logs and Meta Data. In *Proceedings of the ACM Recommender Systems Challenge 2019 Workshop (RecSys Challenge '19), September 20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3359555.3359561

1 INTRODUCTION

Personalized item suggestions that are automatically provided by search or recommendation system are a common feature of many of today's websites and mobile applications. Besides common application scenarios like e-commerce or online media streaming services, e.g., on Amazon or YouTube, also other domains profit from the application of such systems, which showed to increase customer satisfaction and at the same time help businesses to increase their revenue [4–6, 8, 10, 14]. In this work, we focus on search and recommendation problems that are common on hotel booking platforms. These booking platforms play a major role in today's multi-billion dollar tourism industry, which is why also small improvements in the provided service can have substantial business impacts.

Traditionally, hotel recommender systems [2, 7, 9, 12] as well as personalized search approaches [11] often rely on information about the long-term preference of the users, e.g., based on Collaborative Filtering (CF) techniques. However, CF approaches have their limitations in particular in situations of user- and item *cold start*, i.e., when only little information is available about an individual user or when a recommendable item is new on the platform.

In our application domain of hotel search and recommendation, the problem is often particularly severe. Users are not logged in or are first-time users, which means that an adaptation to the user's assumed *intent* and preferences can only be based on the often very few observed recent interactions within the user's ongoing browsing session. This problem, mostly referred to as session-based recommendation, received increased interest in recent years and led to the development of a number of novel technical approaches, both simple and complex ones [17, 19–21]. However, as we will discuss later in detail, these approaches—at least when used in isolation—do not work particularly well for our problem. In such situations, a more promising approach is to rely on hybrid techniques, which consider additional information about items and users [1, 25].

In this paper, we adopt different strategies from the literature and build a novel integrated hybrid method for ranking accommodations according to the user's assumed short-term intents and preferences. Specifically, we capture collaborative signals with the help of matrix factorization as well as neural techniques, address user- and item cold-start through feature engineering, and finally use decision trees to determine the probability that a user is likely to book an accommodation. Technically, we use a hybrid combination of Bayesian Personalized Ranking (BPR) with matrix factorization [22], Doc2Vec [16], and gradient boosting decision trees (GBDT).

The proposed method was designed and successfully evaluated in the context of the 2019 ACM RecSys Challenge. Despite limited resource requirements, it led to the fifth place in a competition with more than 500 teams, with accuracy results that were only 0.53% worse than those of the winning team.

Table 1: Different types of interactions.

Type	Description	Reference
destination	Search for a destination.	Location
info	The user requested an asynchronously loaded item info block.	Accommodation
image	The user opened the hotel's image gallery or clicked on one of the images.	Accommodation
rating	The user requested an asynchronously loaded item rating block.	Accommodation
deals	Request to an asynchronously loaded box with price information from multiple booking sites.	Accommodation
sort	Sorting the hotel list, e.g., by price, distance, or popularity.	Sort option
filter	Filtering the hotel list, e.g., by the minimum number of stars or an important feature like WiFi.	Filter option
poi	The user indicated an interest in a specific point of interest.	POI
search	Search for a specific hotel.	Accommodation
click	The user clicked on an item and, hence, will be redirected to an external booking site.	Accommodation

2 PROBLEM DESCRIPTION

2.1 Computational Task of the Challenge

The computational problem in the 2019 ACM RecSys Challenge hosted by *trivago*[1] was as follows. Given the first interactions of a user's browsing session and a set of 25 hotels that were chosen by some algorithm for display, rank these hotels in a way that the ones with the highest probability of being clicked are placed in the top ranks. The main inputs to the task are:

a) a chronological list of interaction events of different types,
b) a list of hotels to be displayed, and
c) meta data for the user, the hotel, and the context of the event.

The task can be seen as a learning-to-rank problem. For a given list of hotels, learn to rank the relevant (actually clicked) items over negative (not clicked) examples given the previous interactions. The problem setting is different from common recommendation scenarios, where the problem often includes finding suitable items instead of just re-ranking a few. The specific challenges can be summarized as follows.

- *Multiple interaction types.* Considering log events of various different interaction types.
- *Reminding.* Determining whether or not and in which situations users will click on a hotel they previously interacted with.
- *Cold-start.* Ensuring good recommendation quality also for both new users and new hotels.

2.2 Data

The provided dataset consisted of interactions collected over 8 days and meta data about the users, the accommodations, and the context of the interactions. Each log entry included a user and session identifier, the position in the session, a timestamp, an interaction type, and a reference that links the event to an entity, e.g., a hotel for a click or a sorting option for a sort event (see Table 1).

Additionally, each click event was linked to the displayed accommodations alongside their corresponding prices. As *trivago* is an international platform and supports multiple channels, each log entry provided information about the user's country and device type. The logs were collected in several countries and across three devices types, i.e., desktop, tablet, or mobile. This information was particularly interesting as users from different countries presumably have different preferences, especially regarding the price. Inspecting the website on different device types reveals that the amount of information visible to user in the list can differ substantially, which might influence the user behavior.

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

Finally, the dataset contained meta information for each accommodation. Each item was characterized by a set of attributes, e.g., if the accommodation has a satisfactory rating, WiFi, or a pool. Overall, up to 212 attributes were provided per accommodation. For more information about the dataset and the task, see [15].

Table 2: Dataset characteristics

Characteristic	Competitive	Local	Sample
Users	948,014	730,803	430,668
Sessions	1,202,117	910,732	504,141
Accommodations	905,289	853,540	740,119
Interactions	19,715,327	15,932,992	8,782,508
Actions per Session	16.401	17.495	17.421
Interactions per Item	19.237	16.571	10.545
First Time Visitors	78.86%	80.24%	85.43%

2.3 Evaluation Procedure

All evaluations were performed in offline experimental settings. The interaction log data was provided as a training and a test part (covering 6 and 2 days).

2.3.1 Competitive Evaluation. The data points for the "competitive" evaluation were provided in a way that parts of the test set were hidden. Specifically, for each user's last session, the last hotel identifier was hidden in the last click event. Thus, a rearranged list had to be created for the hotels displayed at the time of these click events. The re-ranked lists had to be aggregated in a solution file, which was then uploaded to a special website. After submission, the provided file was automatically evaluated by the system on half of the test set in terms of the metric mean reciprocal rank (MRR). The results of the top teams were displayed on a public leaderboard. At the end, the final leaderboard was determined by evaluating the solutions on the full test set.

2.3.2 Local Evaluation. Uploads to the system were restricted to one submission every 12 hours. Thus, a proper local evaluation environment was important, e.g., for feature testing or selection and parameter optimization. As the competitive test set was created by time-based splitting, we similarly chose the last 1.5 days of the training data as our *local* test set. Again, following the official set, we hid the last clicked reference for each user. Subsequent interactions were ignored. Correspondingly, each user's last click in the training part served as one positive example and several negative examples (non-clicked items in the displayed results). Additionally, to test new ideas, we created a *sample* from half of the *competitive* training data (3 days). The same splitting was applied (25% for testing). Both samples reflected the results of the *competitive* evaluation well. The key characteristics of the datasets are shown in Table 2.

3 TECHNICAL APPROACH

Due to particularities of the problem settings and the enormous sparsity of the data, typical session-based techniques were not suitable for the task. We thus followed ideas from [18] and [24]. Specifically, to determine the desired ranking we framed the problem as a pair-wise learning-to-rank task and engineered a multitude of features based on the log data. Besides several simple statistics, some features were based on the application of more sophisticated techniques. Finally, we trained our models with *LightGBM*[2] due to its small memory footprint and computational efficiency [13].

3.1 Feature Engineering

We designed the following groups of features, which are briefly summarized here. The final list of 518 features can be found online alongside the source code[3].

3.1.1 Basic Features. Some attributes could be fed into the GBDT almost directly. Examples include the platform, the device, the city, the last sort reference, or the position and price of the item.

3.1.2 Features based on Simple Statistics. In addition, we first created simple statistics based on the user, session, and item as well as other given basic features. The features were calculated for individual identifiers or categories and also for pairs and triplets of those, e.g, the number of clicks for an item performed by the user. Furthermore, some of the statistics refer to one or multiple interaction types, e.g., the mean price for an item click for users from a specific country. The statistics can be categorized in 4 groups:

(i) Item. This group focuses on the item to capture popularity information in the dataset, e.g. the number of views, image views, or clicks per platform, per device, or overall. Furthermore, we captured the probability of an item click under certain circumstances, e.g., the ratio of clicks per impressions.

(ii) Price. We mostly determined mean values like, e.g., the mean price of clicked items in a certain city. These features then again were used to calculate a difference or a ratio, e.g., the difference to the mean price clicked on the platform.

(iii) Session. Here, we captured general statistics, e.g., the overall number of events, the number of events of a certain type, or the time spent in a session. Furthermore, we modeled the importance of items within a session by counting their number of occurrences in various ways, e.g., the absolute count, the ratio regarding the session length, the sum when applying a linear decay function to the session interactions, or the overall dwell time in the session.

(iv) User. The last group captures the users' general preferences, mainly regarding the price, e.g., the average price of items a user clicked on (normalized by the city mean). Furthermore, we also captured the users' interest in an item before the current session.

3.1.3 Latent Features. In the given problem setting, we considered all item interactions and the user clicks as implicit feedback signals to create multiple session-item "rating" matrices. This allowed us to derive different latent vector representations for sessions and items. We tested several matrix factorization (MF) techniques for implicit feedback datasets, and ultimately relied on Bayesian Personalized Ranking (BPR)[4] [22]. In an additional approach, we assumed a session to be a sentence, an item to be a word, and applied Doc2Vec to generate alternative lower-dimensional representations[5][16]. For all representations, we calculated the cosine similarity for session-item pairs as a score and included these in our set of features. Due to memory limitations we did not include the latent representations themselves as features, which showed to be effective in [18].

[2]https://github.com/Microsoft/LightGBM
[3]https://rn5l.github.io/rsc19/
[4]We used the implementations at https://github.com/benfred/implicit.
[5]https://radimrehurek.com/gensim/models/doc2vec.html

3.1.4 Features based on Hotel Attributes. The 212 unique hotel attributes provided in the dataset could be directly encoded as binary features, e.g., the hotel has a pool or not. However, to reduce the overall number of features, we again applied Doc2Vec and represented the properties as a 16-dimensional vector. The "documents" were shuffled each epoch while training, see [26].

Nevertheless, we still included binary representations for the 50 most popular attributes in terms of the users' filter requests. Tests on the *sample* showed that this did not lead to lower performance than when higher-dimensional representations were used. Finally, we retrieved additional meta data through *trivago*'s public API, e.g., more precise multi-dimensional rating information and the number of pictures of an accommodation.

3.1.5 Location-based Features. The crawled data also included location information (latitude and the longitude). As we assumed that the distance to the city center or a specific POI might be a valuable feature, we furthermore crawled coordinates for those. Then, we calculated the Haversine distance to the city as well as the last selected POI as an additional feature for each accommodation.

3.1.6 Position-related Features. Besides the general relevance signals discussed so far, we also included characteristics that were specific to the setup of the challenge. We considered the positions both of the candidate item and the previously clicked items in the list. More specifically, we modeled a user's scroll direction and position by looking at previous interactions. We then calculated the differences of a candidate item's position to the previous position, assuming that surrounding items are more likely clicked.

Furthermore, we assumed that items are more interesting to the user when they stand out compared to items shown above or below. Thus, we created features based on the price, rating, stars, and distance of up to 8 surrounding items. These included, e.g., the difference to each position above and below and the value of the candidate item in relation to the mean of *n* neighboring items.

Finally, we combined certain features, e.g., in a price-rating ratio, or a price-distance ratio. From those we engineered "rank features" (1 to 25) following all sorting options on the *trivago* website. Finally, all rankings were aggregated in a Borda count scheme to create one overall heuristic ranking for the candidate accommodations.

3.2 Training, Stabilization, and Ensembles

In order to train, tune, and combine our model in an ensemble, we proceeded as follows. In general, we trained all our models using LambdaRank to find an optimal ranking for the accommodations [3]. To prevent over-fitting, we utilized the last 20% of the training set as validation data for early stopping. As the stopping criterion we applied NDCG@25 (500 iterations) and set the maximum number of iterations to 10,000. We mostly used default parameters, with a learning rate of 0.1 and a feature as well as bagging fraction of 0.5 to further reduce over-fitting and speed up training times[6]. Boosting was performed with DART [23], which showed a slightly better performance than traditional GBDTs and random forests. This process was repeated in a *n*-fold cross-validation training procedure to create an ensemble of *n* models. The obtained ranking

[6]See https://lightgbm.readthedocs.io/en/latest/Parameters.html.

scores were averaged, and taken for the final submission. Due to limitations regarding our computing power, we had to restrict *n* to 5. When increasing *n* in our *sample* evaluation, we could however not observe big leaps forward.

4 RESULTS AND OBSERVATIONS

In the end, our team placed fifth among over 500 registered teams. Our final MRR result was only 0.53% lower than that of the winners (0.6792 vs. 0.6854). Note that features based on additionally crawled data were only used for experimental purposes, as additional data was not allowed for the final submission. In the following, we provide additional details about the relative importance of the used features, their impact on the MRR measure, and other observations.

Table 3: Effectiveness of the different tested feature groups in the evaluation for all datasets.

Feature Group	Sample	Local	Competitive
Basic	0.51269	0.51158	−−
Simple Statistics	0.66786	0.67023	−−
Final w/o Position Features	0.67179	0.67334	−−
Final w/o Crawled Features	0.68062	0.68163	0.68081
Final	0.68103	0.68232	0.68389
5-Fold Ensemble	0.68211	0.68354	0.68413
5-Fold w/o Crawled Features	−−	−−	0.68206

Table 3 shows the impact of some feature groups from Section 3.1. We report the results for the *competitive* (when available), the *local*, and also the half-sized *sample* evaluation setup. Generally, all included feature groups increased the MRR score over the basic model. The features based on simple statistics already contributed a lot to the performance (ca. 30%). The best single model including all groups further improved the accuracy by around 2%. To highlight the influence of the crawled features and the position-based features, we report the results for models excluding those groups. With a decrease of about 0.001 in the MRR, the impact of the crawled information was small. In contrast, the position features helped a lot, increasing the score by ca. 0.01. The 5-fold ensemble was consistently helpful, however, the improvements were rather small.

Table 4 shows the 20 most important features of our final model in terms of the number of branches in the decision trees. In general, all top features are simple statistics. The most important one is the time to the most recent action for the same city in seconds. It might be of particular importance as some users perform a double-click instead of a simple click, and thus the click on an item occurs twice. At the same time, the feature captures if the previous actions were performed for a different city. Overall, the top features consist of a balanced mix of time-aware, price-oriented, position-related, and popularity-based features. All 518 features contributed at least a bit. Feature selection techniques led to a decrease in performance.

Due to hardware limitations, we could not include all desired statistical features. Also, the direct usage of the latent representations based on, e.g, BPR [22] would have exceeded our memory for the *competitive* evaluation. Utilizing those therefore represents an area where further accuracy improvements might be achieved.

Unfortunately, we discovered the importance of the position late in the challenge, which left little time for further feature engineering in this regard. Thus, the extension of the list-based features could lead to higher accuracy. Furthermore, at the end of the challenge, we did not have sufficient computational resources to perform fine-grained hyper-parameter tuning.

Table 4: Split importance of the top 20 features in the final model for the *local* evaluation.

Feature	Splits	Feature	Splits
Time to Previous Action in City	6989	Difference to Mean Item Position	3204
Distance to Last Clicked Position	5265	Overall Mean Item Position	3144
Rank by Popularity	4266	Clicks per Impression of Item	3128
Session Max Dwell Time	4206	Time Difference Last Item Interaction	3124
Price per Mean List Price	4075	Price per Mean Item Click Price	3121
Item Click Per Impression	3961	Session Min Dwell Time	3099
Session Time	3818	Rank Borda Count	3049
Last Action Type	3763	Number of Ratings (Crawled)	3030
Price Above	3370	Price per Mean 2 Surrounding	3012
Session Mean Dwell Time	3307	Stars per Mean 8 Surrounding	2990

Generally, the task in this challenge was very interesting but also very specific. Although features regarding the item positions led to significant improvements in terms of the MRR, they do not seem to be very useful in a real-world scenario. At the stage in the user session where the desired ranking is performed, the user is already presented with a list. Even though the predicted order might represent a better ranking of the accommodations, the list should probably not be re-ranked while the user is exploring it. An exception might be when the user explicitly invokes a re-sort operation. In this case, the predictions could immediately contribute to a better ranking of the items. Otherwise, it might rather be irritating and hurt the user experience. Furthermore, on a more general note, it would have been useful to also know about the search parameters, e.g., the travel dates or the number of guests. Providing this information alongside user histories with the task of creating a good initial ranking of hotels at a destination might have been a more interesting and realistic scenario.

5 CONCLUSIONS

We presented an integrated hybrid method to the problem of ranking hotels based on limited information such as the user's recent browsing history, which is a highly relevant problem in practice for search and recommendation scenarios. For the given task, which could be framed as a pair-wise ranking problem, we engineered a multitude of predictor variables, and used GBDTs as a learning method that finally led to competitive results in the 2019 ACM RecSys Challenge for hotel recommendation. Generally, we found the problem setting to be highly relevant for the research community, e.g., in the context of session-based recommendation and search personalization. The particular design of the challenge as a re-ranking task, however, stimulated the inclusion of certain features, which were helpful for the given task, but which might not necessarily be useful in practice.

ACKNOWLEDGEMENTS

We thank the organizers of the 2019 ACM RecSys Challenge and the staff at *trivago* for the opportunity to participate in this interesting and challenging competition.

REFERENCES

[1] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C. Burguillo, Marta Rey-López, Fernando A. Mikic-Fonte, and Ana Peleteiro. 2010. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences* 180, 22 (2010), 4290 – 4311.

[2] Joan Borràs, Antonio Moreno, and Aida Valls. 2014. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications* 41, 16 (2014), 7370 – 7389.

[3] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06)*. 193–200.

[4] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2012. Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study. *Transactions on Interactive Intelligent Systems* 2, 2 (2012), 11.

[5] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. 169–176.

[6] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *Transactions on Management Information Systems* 6, 4 (2015), 13:1–13:19.

[7] G. Huming and L. Weili. 2010. A Hotel Recommendation System Based on Collaborative Filtering and Rankboost Algorithm. In *Second International Conference on Multimedia and Information Technology*. 317–320.

[8] Dietmar Jannach and Gediminas Adomavicius. 2016. Recommendations with a Purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 7–10.

[9] Dietmar Jannach, Fatih Gedikli, Zeynep Karakaya, and Oliver Juwig. 2012. Recommending Hotels based on Multi-Dimensional Customer Ratings. In *Proceedings of ENTER 2012 - eTourism Present and Future Services and Applications*. 320–331.

[10] Dietmar Jannach and Kolja Hegelich. 2009. A Case Study on the Effectiveness of Recommendations in the Mobile Internet. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. 205–208.

[11] Dietmar Jannach and Malte Ludewig. 2017. Investigating Personalized Search in E-Commerce. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference 2017 (FLAIRS '17)*.

[12] Dietmar Jannach, Markus Zanker, and Matthias Fuchs. 2014. Leveraging multi-criteria customer feedback for satisfaction analysis and improved recommendations. *Information Technology & Tourism* 14, 2 (2014), 119–149.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIPS '17)*. 3149–3157.

[14] Evan Kirshenbaum, George Forman, and Michael Dugan. 2012. A Live Comparison of Methods for Personalized Article Recommendation at Forbes.Com. In *Proceedings of the 2012th European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD'12)*. 51–66.

[15] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. 2019. RecSys Challenge 2019: Session-based Hotel Recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems (RecSys '19)*. 2.

[16] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. II–1188–II–1196.

[17] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1831–1839.

[18] Malte Ludewig and Dietmar Jannach. 2018. Could You Play That Song Again? - Reminding Users of Their Favorite Tracks Through Recommendations. In *Proceedings of the WSDM 2018 Cup Workshop*.

[19] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4 (2018), 331–390.

[20] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance Comparison of Neural and Non-Neural Approaches to Session-based Recommendation. In *Proceedings of the 2019 ACM Conference on Recommender Systems (RecSys '19)*.

[21] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. 130–137.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. 452–461.

[23] Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. 2015. DART: Dropouts meet Multiple Additive Regression Trees. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Vol. 38. 489–497.

[24] Maksims Volkovs. 2015. Two-Stage Approach to Item Recommendation from User Sessions. In *Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge)*. Article 3, 4 pages.

[25] Xinxi Wang and Ye Wang. 2014. Improving Content-based and Hybrid Music Recommendation Using Deep Learning. In *Proceedings of the 22nd ACM International Conference on Multimedia (MM '14)*. 627–636.

[26] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data. In *Advances in Information Retrieval*. 45–57.