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.

RecSys Challenge '19, September 20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-7667-9/19/09…$15.00
https://doi.org/10.1145/3359555.3359561

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].

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

[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
[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 ← already very important.	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.

Session-based Hotel Recommendations Dataset: As part of the ACM Recommender System Challenge 2019

Article *in* ACM Transactions on Intelligent Systems and Technology · November 2020

DOI: 10.1145/3412379

CITATIONS
0

READS
193

6 authors, including:

Jens Adamczak
trivago N.V.
12 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Yashar Deldjoo
Politecnico di Bari
74 PUBLICATIONS **636** CITATIONS
SEE PROFILE

Philipp Monreal
trivago
6 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Session-based Hotel Recommendations Dataset: As part of the ACM Recommender System Challenge 2019

JENS ADAMCZAK, trivago N.V., Germany
YASHAR DELDJOO, Polytechnic University of Bari, Italy
FARSHAD BAKHSHANDEGAN MOGHADDAM, Karlsruhe Institute of Technology, Germany
PETER KNEES, TU Wien, Austria
GERARD-PAUL LEYSON and PHILIPP MONREAL, trivago N.V.,Germany

In 2019, the Recommender Systems Challenge [17] dealt for the first time with a real-world task from the area of e-tourism, namely the recommendation of hotels in booking sessions. In this context, we present the release of a new dataset that we believe is vitally important for recommendation systems research in the area of hotel search, from both academic and industry perspectives. In this article, we describe the qualitative characteristics of the dataset and present the comparison of several baseline algorithms trained on the data.

CCS Concepts: • **Information systems** → **Recommender systems**; **Personalization**;

Additional Key Words and Phrases: Dataset, session-based recommender systems, context-aware recommender systems, tourism, hotel recommendation

ACM Reference format:
Jens Adamczak, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Peter Knees, Gerard-Paul Leyson, and Philipp Monreal. 2020. Session-based Hotel Recommendations Dataset: As part of the ACM Recommender System Challenge 2019. *ACM Trans. Intell. Syst. Technol.* 12, 1, Article 1 (November 2020), 20 pages.
https://doi.org/10.1145/3412379

1 INTRODUCTION

The rapid development of information and communication technologies and the web have transformed the tourism and travel domain. Today, travelers no longer rely on travel agencies but search for information themselves and compose their trips according to their specific preferences. Users have to choose from a multitude of options and recommender systems for travel and tourism can be practical tools to overcome the inevitable information overload. Such developments in e-tourism

Authors' addresses: J. Adamczak, G.-P. Leyson, and P. Monreal, trivago N.V., Kesselstraße 5 - 7, 40221 Duüsseldorf, Germany; emails: {jens.adamczak, gerard-paul.leyson, philipp.monreal}@trivago.com; Y. Deldjoo, SisInf Lab, DEI, Polytechnic University of Bari, Via Orabona 4, 70125 Bari, Italy; email: yashar.deldjoo@poliba.it; F. B. Moghaddam, University of Bonn, Regina-Pacis-Weg 3, 53113 Bonn, Germany; email: farshad.bakhshandegan@uni-bonn.de; P. Knees, TU Wien, Institute of Information Systems Engineering, Favoritenstraße 9-11/194-1, 1040 Vienna, Austria; email: peter.knees@tuwien.ac.at.

2157-6904/2020/11-ART1 $15.00
https://doi.org/10.1145/3412379

have been studied at the ACM Conference on Recommender Systems (RecSys)[1] and the RecSys Workshop on Recommenders in Tourism[2] series, for example.

Recommending hotels and other travel-related items is still a difficult task, as travel and tourism is a very complex domain. Planning a trip usually involves searching for a set or package of products that are interconnected (e.g., means of transportation, lodging, attractions), with rather limited availability, and where contextual aspects may have a major impact (e.g., time, location, social context). Users book much fewer hotels than, for example, listen to music tracks, and, given the financial obligation of booking a stay at a hotel, users usually exhibit a strong price sensitivity and a bigger need to be convinced by any given offer. Besides, travelers are often emotionally connected to the products and the experience they provide. Therefore, decision making is not only based on rational and objective criteria. As such, providing the right information to visitors to a travel site, such as a hotel booking service, at the right time is challenging. Information about items such as hotels is often available as item metadata. However, usually, in this domain information about users and their goals and preferences is harder to obtain. Systems need to analyze session-based data of anonymous or first-time users to adapt the search results and anticipate the hotels the users may be interested in.

Recommendation systems for hotels and accommodations exist in different forms and scopes of application [3]. In this article, we introduce a new public dataset of real-world hotel search sessions released by trivago.[3] We analyze the dataset and present descriptive statistics to highlight the information contained. We additionally compare how several baseline algorithms perform when trained on the data. This article focuses on the description of the dataset used in the RecSys Challenge 2019. The RecSys Challenge is an annual competition that is held in conjunction with the ACM Conference for Recommender Systems. In each challenge, an industrial partner presents a dataset as well as a problem statement to the public. To get a more intuitive understanding of this particular example of the applicability of the data, we will briefly introduce the task posed to participants of the 2019 challenge.

The dataset presented here is tracking interactions of users with the trivago website. Users can search for accommodations, destinations, or points of interest (POIs) and get recommendations presented in the form of a list of results. The users have the option to interact with accommodations shown in the result list by clicking on the content of an accommodation or by making a click that forwards to a booking site (i.e., a clickout). Users can leave the trivago website and return at a later point in time resulting in a new session. The task of the challenge is to use all the information about the behavioral, time-dependent patterns of the users and the content of the displayed accommodations to develop models that allow to predict which accommodations a user is most likely to click on when presented with a list of potential options. More details about what kind of information is available in the data can be found below in Section 3.

2 EXISTING DATASETS

Over the past few years, multiple datasets have been collected to aid the development of recommender system solutions for practical applications. However, domain-specific datasets that are relevant for the travel industry typically focus on descriptive content of information that can be scraped from a website. Other datasets that focus on user responses in recommendation settings, such as datasets from previous RecSys challenges, provide more information about website usage patterns but have so far not allowed drawing conclusions about the behavior of users in a

[1] https://recsys.acm.org.
[2] http://www.ec.tuwien.ac.at/rectour2019/.
[3] https://www.trivago.com.

Table 1. Overview of Comparable Datasets Created for the Development of Recommender Systems, as Well as the Dataset Presented in This Paper (Shown Below in Bold)

Dataset	Source	Row type	Description	Row count
BCOM19	Booking.com	review	reviews, scores, metadata (tag)	515K
DF19	Datafiniti	hotel	reviews, rating, and metadata	35K
GB19	goibibo.com	review	reviewsand rating	4K
MMTRIP19	MakeMyTrip.com	hotel	reviews and rating	20K
TRIP09	TripAdvisor	hotel	reviews and rating	235K
RS15	Yoochoose	session	clicks and purchases	33M
RS16	Xing	user	impressions and interactions	1B+8.8M
RS17	Xing	user	impressions and interactions	314M+7M
RS18	Spotify	user	playlists	1M
RS19	**trivago**	**session action**	**various types of user actions**	**19.7M**

travel-focused environment. With this article and the associated dataset, we aim to bridge the gap between the different types of information that are currently available and provide relevant information for the development of recommendation systems in the online travel domain.

This section describes datasets from both scenarios, i.e., data from the travel domain, as well as the datasets that capture the user response in recommendation settings as presented in the previous RecSys challenges. These datasets are compared to the one presented in this article from a structural perspective. A brief overview of the different datasets is shown in Table 1.

2.1 Domain-specific Datasets

Most of the existing datasets in the travel domain pertain to reviews or ratings of different hotels. Some of these datasets are listed below. These datasets differ from the dataset presented in this article in the sense that they do not contain any information about actual user interactions on travel websites. Rather, they describe metadata and content of hotel inventory and location information. We aim to complement this static information with a more dynamic description of interaction patterns that illustrate how users are responding to content that is provided for them.

BCOM19 [5]: This dataset consists of 515K customer reviews and scores of 1,493 luxury hotels across Europe. It also supplies fine-grained information about reviews/reviewers, e.g., if the reviews are positive/negative, then the total number of reviews, the nationality of reviewers along with tags assigned by reviewers to hotels, and hotel location (longitude and latitude). The data were scraped from Booking.com.

DF19 [10]: This dataset provides a list of 1,000 hotels and 35K reviews provided by Datafiniti's Business Database. The dataset supplies additional metadata such as hotel location, hotel name, rating score, review data, title, username, and so on.

GB19 [12]: This dataset supplies a subset, i.e., 4K, of a bigger dataset extracted from goibibo.com, a leading travel portal in India. It contains a wide range of metadata, such as information specific to hotels (category, description, facilities, star rating, image count), location, point of interest, and others. The original dataset contains information on 33K hotels.

MMTRIP19 [21]: Similarly to *GB19* [12], this dataset is a smaller version of a large dataset taken from MakeMyTrip.com, a major travel portal in India. This dataset also includes metadata information (hotel overview, star rating, image rating), location, and rating. The original dataset contains information on 615K hotels and is available on DataStock, a data repository website supplying a historical record of several industries.

TRIP09 [28]: This dataset contains hotel reviews from TripAdvisor in a month (from February 14, 2009, to March 15, 2009). Besides the overall rating, this dataset contains additional information such as aspect ratings in each review: value, room, location, cleanliness, check-in/front desk, service, and business service ranging from 1 star to 5 stars.

2.2 Previous RecSys Challenge Datasets

Since the currently publicly available datasets from the travel domain lack information about user responses, they can not be used to build and evaluate recommendation algorithms. To make the reader aware of more similar datasets that have been successfully used in the past to evaluate novel recommendation system approaches, we offer a comparison with datasets from past RecSys challenges. The last four RecSys challenges are described below.

RS15 [4, 32]: This dataset comprises of session-based clicks and purchases of users using an e-commerce website. The task given by the industry partner is to predict whether a user is going to buy an item or not. If a purchase is predicted, then a prediction of which item the user buys is also required. This dataset was provided by Yoochose.

RS16 [1, 30]: This dataset contains user-based impressions and interactions using a job posting website (XING). Metadata about the job postings are also provided. The task for this dataset is to predict which job postings are likely to be relevant to the users. This dataset was provided by Xing.

RS17 [2, 31]: This dataset is similar to RS16, but both the dataset and task are modified to include a commercialization dimension. Participants are asked to balance relevance and revenue, as well as tackle the novelty/sparsity of information about new job postings. This dataset was also provided by Xing.

RS18 [6, 14]: This dataset contains user-based playlists by users of a music streaming website. Metadata about the tracks are also provided. The task is to generate an automatic playlist continuation based on the users' existing playlists [25]. This dataset was provided by Spotify.

The dataset presented in this article (RS19) consists of anonymous session actions of trivago users and was opened to the public in the RecSys Challenge 2019 competition [17]. These actions not only included impressed/clicked hotels but also other website interactions, such as filter usage or search types. This introduces a different dimension to the existing datasets in the travel domain, which tend to be more centered around accommodations and their properties. In comparison to the previous RecSys challenges, our dataset introduces a finer granularity of user interactions with a website.

3 DATASET DESCRIPTION

This section summarizes the characteristics and data structure of the data and highlights some interesting features of it.

3.1 Overview

The dataset consists of the sequential website interactions of users visiting the trivago website between November 1, 2018, and November 8, 2018, as shown schematically in Figure 1. The data contain a wide variety of user interactions that include, for instance, making a click that forwards to a booking site (i.e., a clickout), an interaction with an item image, or a selection of a filter. An overview of all interaction types that are contained in the data can be found in Table 2.

Each website interaction corresponds to a specific timestamp in the dataset. Multiple website interactions can appear in a user session. A session is defined as all interactions of a user on a

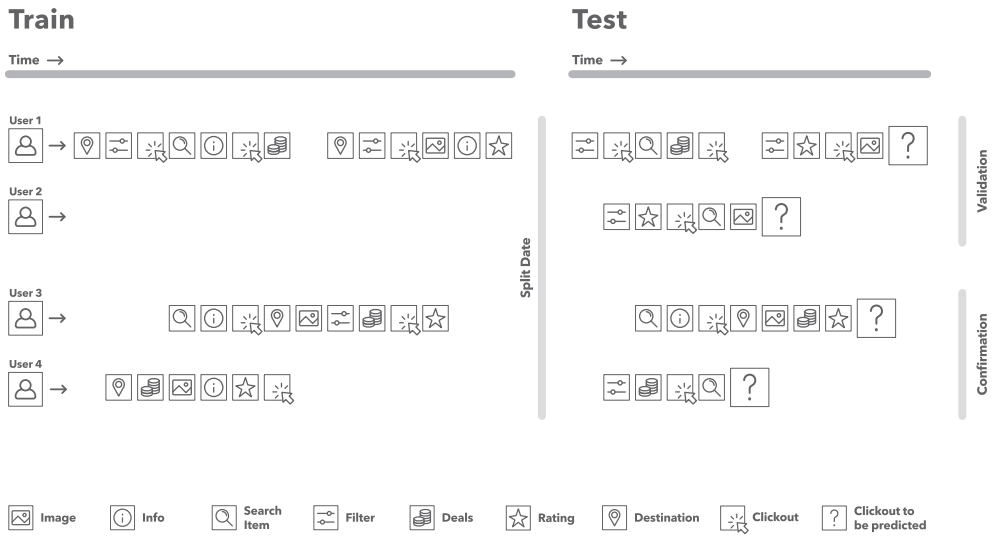

Fig. 1. Schematic illustration of the dataset and how it is split between train and test set. Each icon in the schematic represents a different type of website interaction, such as clicking on item content (image, info, rating, deals), refining the search parameters via filtering, or triggering new searches for accommodations (items) or destinations. All interactions that can be performed by the users and that are indicated by the icons in the schematic are in more detail described in Table 2. Gaps between consecutive interactions indicate the start of a new user session. The train set contains sessions before November 7, 2018, while the test set contains sessions after said date. The item_id of the final clickout (shown as the box with the question marks) have been withheld. Note that the question mark refers only to the accommodation identifier that needs to be predicted and not the action type and that every event for which a prediction needs to be made is a clickout. For the evaluation of the leaderboard, the test set has been split into a confirmation and a validation set on a user basis.

Table 2. Description of Action Types and Reference Values for All Possible Session Interactions

Action type	Reference	Description
clickout item	item id	Click on item and forwarding to a booking site
interaction item rating	item id	Interaction with a rating or a review of an item
interaction item info	item id	Interaction with item information
interaction item image	item id	Interaction with an image
interaction item deals	item id	Click on the *View more deals* button
change of sort order	sort order name	User changes the sorting order
filter selection	filter description	Filter selection, e.g., four stars
search for item	item id	Search for the name of an accommodation
search for destination	city name	City search, e.g., Austin, TX
search for POI	POI name	Point of interest (POI) search

specific trivago country platform with no gaps between the interaction timestamps of >60 minutes. If a user stops interacting with the website and returns after a couple of minutes to continue the search, then the continued interactions will still be counted to belong to the same session. Because of the grouping of website interactions into sessions, the interactions are in the following often referred to as session actions. For each session interaction, data about the context of the interaction are provided, e.g., the country platform on which the interaction took place or the list of items that

Table 3. General Statistics of the RecSys Challenge 2019 Dataset

	Training dataset	Test dataset	Item metadata
file size (MB)	2,100.81	534.91	257.90
row count	15,932,992	3,782,335	927,142
column count	11	11	2
unique sessions	910,683	291,381	—
unique users	730,803	250,852	—
clickouts	1,586,586	528,779	—
min. date	2018-11-01	2018-11-08	—
max. date	2018-11-07	2018-11-09	—
avg. sessions per user	1.246	1.162	—
avg. actions per user	21.802	15.078	—

were shown at the moment of a click and the prices of the accommodations. Metadata for each of the accommodations are provided in a separate file.

3.2 File Descriptions

The dataset used in the challenge consists of three files: train.csv, test.csv, and item_metadata.csv. General statistics of these files are summarized in Table 3. The first two files contain the session actions of various uses, while the last file contains information about item amenities that can be cross-referenced for additional information.

In addition to these three files that were made public to the participants in the course of the challenge, we make available the ground-truth data used for evaluating the submissions. The ground-truth data consists of a validation.csv file used for the calculation of the public leaderboard, and the confirmation.csv file used for the calculation of the final leaderboard. Both files are a subset of the test_ground_truth.csv file that contains the information of accommodation that has been clicked. The files can be accessed from a dedicated website.[4]

The following dataset descriptions focus on the three public datasets.

3.2.1 Session Actions Files. Each row in the session action files (train.csv and test.csv) corresponds to a particular user action in a given session. The schema of these files is shown in Table 4. The split between the train and test sets were done at a particular split date. That is, sessions that occurred before November 7, 2018, were put into the train set, while those that occurred after were put into the test set. The target of the test set are items clicked out at the end of the sessions in the test set.

A user can perform a wide range of actions to interact with the item list. Depending on the action type, the "reference" column could represent different things, such as an item identifier or a city name. For instance, if a user interacts with the rating of an item, then the reference value contains the identifier of the item that was interacted with. A list of those actions is summarized in Table 2.

In addition to the user actions, the train.csv and test.csv files also contain information about the accommodations that were displayed to the user at the time a clickout was made. An accommodation that is displayed is referred to as being "impressed" and all displayed accommodations are stored in the "impressions" column. Each row in that column is a list of accommodations (items) in the order in which they were displayed on the website. In case the user action was not a clickout, the impressions column is left empty.

[4] https://recsys2019data.trivago.com/.

Table 4. Column Descriptions of the Session Action Files (train.csv and test.csv)

Column name	Data type	Description
user_id	String	Identifier of the user
session_id	String	Identifier of the session
timestamp	Timestamp	UNIX timestamp for the time of the interaction
step	Integer	Step in the sequence of actions within the session
action_type	String	Description of the user action at this session step
reference	Integer	Reference value for the different action types
platform	String	Country platform that was used for the search
city	String	Name of the destination of the search context
device	String	Device that was used for the search
current_filters	String	Pipe-separated list of filters that were active at the timestamp
impressions	String	Pipe-separated list of items that were displayed to the user
prices	String	Pipe-separated list of prices of impressed items in Euro

Table 5. A Sample Row of the Session Actions File (train.csv and test.csv)

Step	action_type	Reference	Current_filters	Impressions	Prices
1	search for destination	Barcelona, Spain	—	—	—
2	filter selection	Focus on Distance	Focus on Distance	—	—
3	search for POI	Port de Barcelona	Focus on Distance	—	—
4	interaction item deals	40255	—	—	—
5	clickout item	40255	—	6744|40181|...	162|91|...
6	search for item	81770	—	—	—
7	interaction item info	81770	—	—	—
8	clickout item	81770	—	6832|40396|...	347|245|...

To illustrate the different session actions more concretely, Table 5 shows an example of actions performed in a session by a user from the US platform has used trivago on a desktop device. The actions in this session are the following:

(1) User searches for Barcelona, Spain (action type: search for destination, reference: Barcelona, Spain).
(2) The "focus on distance" filter is activated. At this point, the current_filters column indicates that this is the only filter that is active (action type: filter selection, reference: Focus on Distance).
(3) User searches for a POI, the Port de Barcelona (action type: search for POI, reference: Port de Barcelona).
(4) User viewed at the "More Deals" button on item 40255. The "focus on distance" filter is no longer activated (action type: interaction item deals, reference: 40255).
(5) The user clicks out on item 40225. The full list of displayed items and their associated prices can be seen in the "impressions" and "price" columns (action type: clickout item, reference: 40225).
(6) User searches for item 81770 (action type: search for item, reference: 81770).
(7) User interacts with the item information of item 81770 (action type: interaction item info, reference: 81770).
(8) User clicks out on item 81770. The full list of items and their associated prices can be seen in the "impressions" and "price" columns (action type: clickout item, reference: 81770).

Table 6. Column Description of the Item Metadata File (item_metadata.csv)

Column name	Data type	Description
item_id	Integer	Identifier of the accommodation. Used in the reference values for item related action types.
properties	String	Pipe-separated list of properties of items

things to account in EDA.

The dataset contains detailed information about session interactions and at the same time exhibits sparsity and imbalances in certain dimensions that make it challenging to derive personalized and context-specific recommendations based on previous interactions. About a third of the users (33.6%) did not interact with any items before making the final clickout. The top 10 locales constitute 62.3% of the total actions. The majority of actions (74%) are interactions with the item image.

3.2.2 Item Metadata. Apart from the session action files, a separate file containing the item metadata is provided in the dataset ("item_metadata.csv"). The column descriptions of this file are described in Table 6. Item properties can be the classification of the accommodation (e.g., the hotel star rating, bed-and-breakfast or "serviced apartment"), amenities of the accommodation (e.g., balcony, free wifi, swimming pool) or available services (e.g., laundry service, massage). There are 157 unique properties in the dataset.

This dataset can enrich the previous datasets for actions relating to items or lists of items by referencing these items using the item_id. There is a large imbalance in the number of properties per item. The mean number of properties is 19.7, but 20% of the items have only 1 property.

3.3 Dataset Characteristics

Before exploring the structure of the dataset in the context of building predictive models in Section 4, we discuss some of the characteristic properties of the data. The data are a collection of user interactions across 55 country platforms. It allows us to inspect individual user preferences in terms of selected destinations and accommodations, and the analysis of website usage patterns on platform level.

3.3.1 User Search Preferences. A direct way for a user to specify a preference is to search on the website for a destination, a POI, or an accommodation. This information is contained in the dataset in the form of search-related action types. The most commonly searched destinations and POIs are displayed in Table 7.

User preferences specified in this way are useful in the development of recommender systems. In reality, information about these preferences is often sparse. This is also true to a certain extent for the dataset presented here. The number of searches for the individual destinations and POIs is small compared to the large number of total interactions that are registered for the top platforms, as can be seen in Table 8. Searches for destinations and POIs combined make up for only 3.4% of all interactions. The search for an item that would be very valuable in the context of the challenge as an explicit signal of interest in a particular accommodation, accounts for only 0.96% of all interactions. Directly specified user preferences alone are not sufficient to reliably predict clicked accommodations. The explicit search information needs to be complemented by the other, implicit, session actions that capture the website usage patterns and that are more frequently performed.

3.3.2 Website Usage Patterns. Usage patterns can be described by the temporal dependency of user interactions. Figure 2 displays the number of interactions per world region for the training and test time frame of the dataset. The world regions group the different trivago platforms by

Table 7. Top 10 Searched Destinations and POI in the Training Dataset

Rank	City	Searches	(%)	POI	Searches	(%)
1	London, United Kingdom	6,421	1.59	Tokyo Disneyland	2,093	1.52
2	Paris, France	4,858	1.21	Kyoto Station	1,303	0.95
3	New York, USA	4,667	1.16	Las Vegas Strip	1,238	0.90
4	Amsterdam, Netherlands	3,609	0.90	Tokyo Station	1,233	0.90
5	Las Vegas, USA	2,960	0.73	Shinjuku Station	1,114	0.81
6	Berlin, Germany	2,764	0.69	Melbourne CBD	1,099	0.80
7	Barcelona, Spain	2,686	0.67	Times Square	1,082	0.79
8	Rome, Italy	2,615	0.65	Sydney CBD	974	0.71
9	Cancun, Mexico	2,460	0.61	Osaka Station	960	0.70
10	São Paulo, Brazil	2,357	0.58	Hakata Station	927	0.67

The preferences are given in the absolute number of searches and in the percentage of searches that a particular city or POI received relative to the total number of city or POI searches.

Table 8. Top 10 Trivago Country Platforms in Terms of Total Interactions That Users from That Platform Had with the Website in the Training Dataset

Platform	Country	Interactions	(%)
BR	Brazil	2,634,304	16.53
US	USA	1,627,520	10.21
DE	Germany	1,001,105	6.28
UK	United Kingdom	918,900	5.77
MX	Mexico	833,785	5.23
IN	India	679,747	4.27
AU	Australia	595,003	3.73
TR	Turkey	564,271	3.54
JP	Japan	547,480	3.44
IT	Italy	527,046	3.31

The interactions in the top 10 are relatively evenly distributed across the different platforms with Brazil and the USA standing out with a higher share of overall interactions.

similar time zones according to geographic regions in the United Nations M49 standard. The regions are reduced to three geographic areas, America (North America, and Latin America and the Caribbean), Asia and Oceania, and Europe and Africa. There are clear hourly variations due to the normal seasonality of common human daily routines, i.e., less usage of the website late at night for the respective time zones. Apart from the daily seasonality, there are differences between the different days of the week. The time series for Europe and Africa shows a more complex profile that potentially can be decomposed into more specific user patterns for the individual platforms that constitute these world regions. There seem to be geographical differences between the regions.

The frequency of the type of interactions that are performed is shown in Table 9. The most common website interaction is the interaction with the image of an accommodation on the result list page. It accounts for 74% of all interactions. The next most common interaction is the clickout on the item with almost 10% of the total number of interactions. In addition to their overall frequency in the dataset, interactions with an image of an item are useful indicators to infer what accommodation is eventually clicked as they are often directly followed by a clickout to the same item.

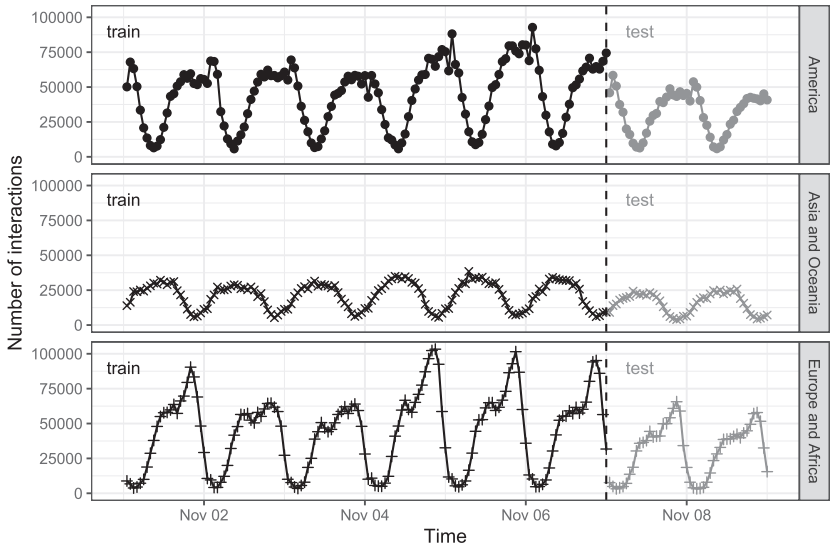

Fig. 2. Temporal patterns of user interactions for the different world regions. The Time axis displays the date and time in Universal Time Coordinated (UTC). Interactions happen in a pattern of daily seasonality that has different peaks depending on the current timezone and daily browsing preferences.

Table 9. Top 10 Trivago Country Platforms in Terms of Interactions in the Training Dataset

Action type	Occurrence	[%]
interaction item image	11,860,750	74.44
clickout item	1,586,586	9.96
filter selection	695,917	4.37
search for destination	403,066	2.53
change of sort order	400,584	2.51
interaction item info	285,402	1.79
interaction item rating	217,246	1.36
interaction item deals	193,794	1.22
search for item	152,203	0.96
search for POI	137,444	0.86

Figure 3 shows the frequency of action types that are the direct precursor of a clickout event. The direct precursor action is the action that happened at the step directly before a clickout in a session of a user. The most common event that happens directly before a clickout is another clickout. This is the case for about 30% of all clickouts. In 21% of all cases, the clickout before happened on another item than the item that was eventually clicked. In 9% the clickout goes to the same item. The action types that are the next likely to happen right before a clickout is the interaction with the image of an item. This is not surprising as the image interaction is the most common action type overall. It is still interesting to see that for the interaction with an image of the item it is more likely that the eventual clickout goes to the same item than for the clickout action type. The item interaction appears to be a strong indicator of the final clickout choice if it happens directly before the clickout. It is furthermore remarkable that in almost 20% of cases there is not direct action type preceding a clickout, i.e., the clickout is the first action type that is recorded in the session. This

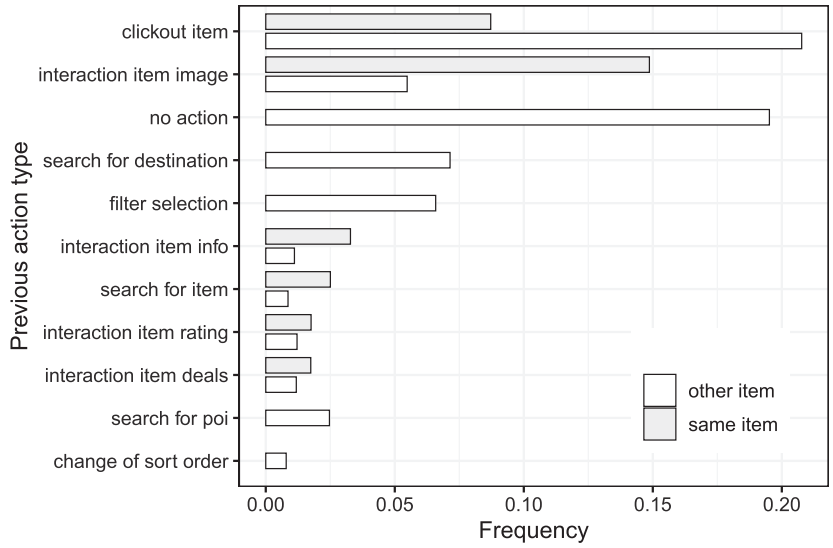

Fig. 3. Overview of the frequency of the action types that directly precede a clickout. The most common event that happens before a clickout is another clickout. Interactions with the image of an item are disproportionately often followed by a clickout that goes to the same item instead of going to another item.

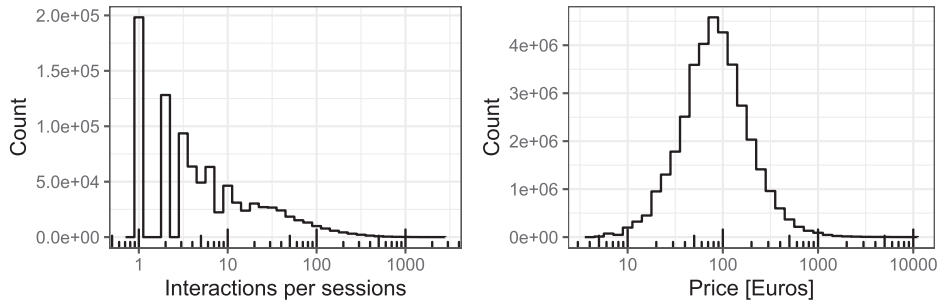

Fig. 4. Distribution of data for the number of interactions per session (left panel) and prices of accommodations in the impression list (right panel) in the training set. Note that the data are presented on a log scale, illustrating the uneven distribution of data. The majority of sessions consists of only one interaction.

makes the development of models based on previous session information challenging and requires the pooling of additional information across sessions and users.

Another difficulty is that the amount of data on interactions and clickouts is rich in aggregation but very unevenly distributed, leading to sparsity on session and user level. Figure 4 illustrates the distribution of data using the example of interactions per session in the training set. The majority of sessions consist of only one interaction. Few sessions are available that have complex user patterns and more than 100 interactions. The sparsity of the data is even more extreme for information about individual users. Of the 730,803 users in the training set, 84% have only one session to account for. To infer the identity of a clicked item, additional information about individual users and their respective browsing history inevitably needs to be complemented by information about the context of the clickout. In that regard, the dataset contains information about over 36 million prices of accommodations that were displayed to the users in the impression lists. A histogram of these prices is shown in the right panel of Figure 4. In contrast to the very sparse session interaction and

user history information, prices are distributed closer to a log-normal distribution, revealing the range of the most commonly displayed prices when presenting the data on the logarithmic scale.

In the next section, we will explore methods that use the information in the data to infer the accommodations of the clickouts that were withheld in the test data.

4 EXPERIMENTS

The dataset can be utilized for a variety of applications. We restrict our exemplary exploration to the context of the 2019 RecSys Challenge problem formulation. The challenge is essentially a ranking task. Participants have to take the list of displayed accommodations in the impression list and submit a list of items in a new order according to the item click propensity for each missing clickout. In this section, we present a set of baseline algorithms that illustrate how to train models on the data and calculate predictions. The selected models draw inspiration from the context of the challenge and highlight specific aspects of the dataset.

Sophisticated algorithms exist that aim at exploiting the session-based nature of data to predict user preferences and provide recommendations, e.g., GRU4Rec [13]. We refrain from going into more detail about these kinds of algorithms in the experimentation section for two reasons. First, it has been shown that for many datasets more basic algorithms can achieve competitive performances that are easier to implement and have more moderate requirements on training time and faster inference speed [19]. This is in line with the goal that algorithms, presented here can be run within a short time in a local environment and do not have overly challenging hardware requirements.[5] Second, the dataset presented in this article is peculiar in the way that it inhibits an extreme sparsity of data for individual sessions and users and often short session lengths. This makes it challenging to apply known session-based recommendation systems, and among the top-performing participants of the challenge indeed no team did. Participants rather opted to extract meaningful session-based features and revised methods that drew information across different sessions and users.

In the experimental part of this article, we will focus on benchmarks that describe certain key aspects of the data that turned out to be important to achieve high performance in the challenge. We will contrast the benchmarks with the more advanced state-of-the-art models that were developed in the course of the challenge in Section 4.7. The code to reproduce the experiments with the dataset of the challenge is stored in a publicly available repository.[6]

4.1 Evaluation

We evaluate all presented algorithms with the metric used in the challenge, the mean reciprocal rank (MRR). For a list of ranked items, the reciprocal rank is the multiplicative inverse of the rank of the first positive response. For example, if an item i is clicked or booked on position rank_i in the result list (counting from the top), the reciprocal rank is denoted as $1/\text{rank}_i$. The mean reciprocal rank is the average of all reciprocal ranks for a given number N of inspected results lists,

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}. \tag{1}$$

In addition to the MRR metric, we provide a second performance measure in the form of the average precision@3. Precision is defined as the fraction of relevant items, in this case, clicked, to the user for a given search query. In this concrete example, we measure if the clicked item I_c

[5] All models presented have been run on a 2.7-GHz Intel Core i5 machine with 16-GB 1867-MHz DDR3 ram within the time frame of a few minutes.

[6] https://github.com/trivago/recsys-challenge-2019-benchmarks.

Table 10. Overview of Performance of Baseline Algorithms in Decreasing Order of Performance

Model	MRR		Average Precision@3	
	Validation	Confirmation	Validation	Confirmation
GBMrank	0.647	0.645	0.230	0.229
Logistic Regression	0.642	0.640	0.228	0.227
nn-Interaction	0.634	0.632	0.224	0.223
nn-Item	0.503	0.500	0.182	0.181
Position	0.502	0.500	0.181	0.181
Popularity-users	0.290	0.290	0.103	0.103
Popularity-absolute	0.288	0.288	0.102	0.102
Random	0.177	0.177	0.051	0.051

appeared in the top 3 submitted results. We annotate this by the indicator function $[I_c \in top_3]$. The function is 1 if the item appears in the top results and 0 if it does not. The average precision@3 is the average across all clicks N that have to be predicted,

$$Avg.Precision@3 = \frac{1}{N}\sum_{i=1}^{N}\frac{[I_c \in top_3]}{3}. \quad (2)$$

To evaluate the algorithms the test set is split on a user basis into a validation set and a confirmation set. The performance achieved on the validation set corresponds to the public leaderboard that was available for participants during the challenge. The performance achieved on the confirmation set corresponds to the final performance as evaluated after the end of the challenge.

4.2 Basic Benchmarks

In the following sections, we describe methods build upon the information contained in the training dataset only. The algorithms are afterward applied to the test data to rank the list of impressed items for each missing click based on the relevance of the items determined by each method.

For each clickout, the dataset contains information about the impressed accommodations, i.e., the list of items that were displayed to the user at the time of the clickout. The order of the accommodations in the impression list corresponds to the order in which they were displayed. In other words, the order corresponds to the original sorting of the list as calculated by a baseline algorithm that was used in the production of the dataset. We can make use of this order to measure the quality of this baseline algorithm by not re-ranking the items in the impression list and leaving the order unaltered for the evaluation. We refer to this order as *Position* in Table 10 indicating that the position in which the items appeared originally is responsible for the performance on the evaluation sets. Essentially, this means ignoring any session-based data. The position of the hotel in the original impression list was regularly identified as a top feature in the models of the challenge participants. This benchmark gives an indication of what value for the final prediction this feature has and what results could have been achieved if a submission had relied solely on this feature.

As a second baseline, we introduce the result that can be achieved when submitting a randomly shuffled list. This is not per se a ranking algorithm. The score will be influenced by the typical list length of impressed items and can be regarded as a lower bound that each algorithm should beat. This benchmark is referred to as *Random* in Table 10 and as expected has the lowest MRR and average precision value of all algorithms presented. This benchmark serves as a lower bar for all evaluated options and reveals that there is variability in the length of the impression lists provided

in the dataset. The maximum number of items that appear on the list is 25. If all lists were of the same length, then we would expect an MRR of 0.15, i.e., the mean of the reciprocal ranks from 1 to 25. There must be significant instances of lists in the dataset that are shorter.

4.3 Popularity-based Methods

Popularity-based recommendation algorithms measure the importance of individual items based on how frequently users interacted with them in the past. They cannot account for changes in user interest and are limited in their capability to explore new inventory but they usually provide a reasonable benchmark for other algorithms as they well capture recent user interest.

4.3.1 Absolute Item Clicks. We present two variants of popularity-based algorithms. The first one defines popularity of item i as the absolute number of clicks the item received in the training phase. For the final predictions on the test set, items will be sorted in decreasing number of training clicks. If multiple items in an impression list have not received any clicks in the training phase, then they will be ordered according to their original position. The performance of the popularity-based result can be seen in Table 10 under the name *Popularity-absolute*. The score achieved is lower than the *Position* estimate indicating that the popularity-based method can only provide a very basic benchmark in this scenario.

4.3.2 Distinct Users. The second variant of a popularity-based method characterizes popularity for item i by the number of distinct users that clicked on that item. Again the assessment is made for the training phase and the method is applied to the test set. The performance of this variant (*Popularity-users*) is slightly higher than *Popularity-absolute* as it reduces the influence of repeated clicks on the same items.

The low performance of the popularity-based algorithms illustrates again the fact that the dataset is very sparse. The majority of the predictions that need to be made are for first-time users looking for items with little historic information.

4.4 Nearest-neighbor Algorithms

Nearest-neighbor algorithms calculate predictions by defining a similarity measure between items and for a target item recommend the n closest items in the given similarity measure. They have been shown to deliver satisfactory results for various use cases [26]. We provide results for two versions of nearest-neighbor approaches. For both versions, we identify the last item that a user interacts with in a given session as the target item and calculate the similarities between this item and all items in the impression list. The two methods differ in the way they calculate the similarity measure.

4.4.1 Item Based. For the item-based nearest-neighbor approach, we calculate the similarity based on the item metadata. For this purpose, we construct a vector for each item characterizing the metadata that is available for that item in the item_metadata.csv. More concretely, the vector for each item is a binary vector with length 157, i.e., the number of distinct properties in the item metadataset. Each element e of the vector is set to 1 if an item has property e and is otherwise left to be 0. Once the metadata vectors have been constructed, we identify for each user the last item that was clicked in the test set before the final list of impressed items was presented. Subsequently, we calculate the cosine similarity between the vector of the last clicked item and the vector of each item that appears in the impression list. We then sort the list in decreasing order of similarity, i.e., showing items on top that are more similar to the last clicked item. The item metadata-based method is indicated as *nn-Item* and performs better than all previous benchmarks.

4.4.2 Interaction Based. The interaction-based approach works similarly to the item metadata-based one and uses the same method as described in Reference [13]. A similarity between two items is measured based on the number of session co-occurrences of these items. Concretely, we first construct a binary vector for each item i in the length of the number of sessions in the training data. The value for item i and session s will be set to 1 in case the item appeared in the session, otherwise the value will be set to 0. For for an item to qualify as being present in a given session it needs to have had at least one user interaction in that session. Similarly to the previous case we calculate the cosine similarity of the interaction vector of the last clicked item and each item in the impression list and sort the list accordingly.

The fact that the interaction-based nearest-neighbor algorithm performs so much better than the item-based one illustrates that for the given dataset interaction data appears to be more useful than the content that is provided for the items. Both nearest-neighbor models calculate similarities between the last interacted item and items in the impression list. Especially the interaction-based method (*nn-Interaction* in Table 10) performs surprisingly well and shows that previous session interactions are a strong signal to predict how likely a user is to click on a particular item. The importance of incorporating recent interaction data as a feature into the predictions for a particular user was recognized by the challenge participants and heavily used in the feature engineering process in the development of the models of the top-performing solutions.

4.5 Binary Classification

Some of the participants of the challenge formalized the challenge task in the form of a binary classification problem or a click prediction task. In that definition, a click probability for each item in the impression list needs to be calculated based on different item and session related features. To tackle the challenge in this framework we first transform the training data to restrict it to the clickout action type and expand the impression and price list to have each row represent an impression. We add a column that indicates if a given item has been clicked or not as the target variable. To calculate the final predictions we transform the test data in the same way and apply the model that was trained on the training data. There are multiple options to solve a click prediction problem, including linear models, boosted tree methods, and also certain neural network architectures. A standard model that is used widely in the industry is the logistic regression model. It is known to perform especially well in cases where the signal to noise ratio is low and on smaller datasets [23]. Our dataset is not small but very sparse, which makes it hard to detect the signal.

We select four input item-specific features, the position of the item in the original listing, the price of the item, the number of previous interactions that the user had with a given item, and an indicator if an impressed item is the last item the user interacted with. We train the model on the training set and apply it to the test set data. The items in the impression list for each missing click are sorted in decreasing click probability. With this basic feature set, the algorithm already performs very well as can be seen in Table 10 for entry *Logistic Regression*.

The good performance of the logistic regression model hints at the importance of the selected features. The model itself is basic and the selection of relevant features makes it possible to achieve an acceptable result without any model optimization and parameter tuning.

4.6 Learning to Rank

In the previous example, the relevance of an item was calculated according to its probability of generating a clickout. Depending on the scenario, the absolute click probability of an individual item might be of less importance than the likelihood that more relevant items appear higher in the list than less relevant items. Techniques aimed at learning to rank address this problem by looking

at combinations of items and introducing loss functions that quantify if combinations of items are ranked appropriately.

One particular example of a learning to rank model is LambdaRank. In this approach, a ranking is optimized by considering pairs of items. Model parameters are determined by weighting the gradient of a pairwise loss function by the change in ranking accuracy that occurs when swapping two items. Typically the normalized discounted cumulative gain (NDCG) is used to determine the ranking accuracy. NDCG is equivalent to the MRR in cases in which only a binary outcome is measured and is therefore a fitting optimization measure for our use case.

In our approach we use the LightGBM implementation of the LambdaRank method [16]. The loss function in this implementation consists of gradient boosted trees. We specifically pick LightGBM to calculate the benchmark, because it was chosen by many participants of the challenge. Some of the top-performing solutions used this implementation with slightly different choices of the objective function and boosting method.

The input data for the model are prepared in the same way as for the logistic regression model. We use the same feature set as before as an input for the ranking model. Compared to the logistic regression model, LightGBM needs an additional input that specifies the search query, i.e., the individual result list, that an impression belongs to to identify suitable pairs of items to compare. The size of the search query is equivalent to the length of each impression list and we feed this length into the model.

As opposed to a click probability, the LightGBM model produces a relevance score for each impressed item. As for the previous examples, we sort the items by decreasing relevance before evaluating the results. Table 10 shows the result of this scoring process under the reference GBM-rank. This method outperforms all other tested approaches. If the LightGBM model with the few features introduced here would have been used in the challenge, then it would have ended up in the top 15% of submissions. This illustrates that a careful selection of features in combination with a strong recommendation model is a promising approach for the given data. Unsurprisingly the LightGBM model was the workhorse in the development of a lot of models in the competition.

The described benchmark models only pinpoint some of the characteristics of the dataset. To achieve a better performance, more sophisticated models were developed by the challenge participants.

4.7 Algorithm Performance

We present the results of our benchmark models and contrast them with the results of the state-of-the-art models that have been developed by participants of the challenge.

Table 10 summarizes the results for all tested benchmark approaches. The motivation for choosing a particular benchmark model has been provided for the description of each benchmark. In summary, the presented algorithms were selected to demonstrate the utilization of different features of the datasets that turned out to be relevant in the challenge. They are by no means exhaustive and do not claim to fully explore the characteristics of the data. Individual refinements for each of the chosen methods can potentially yield higher scores, as can the combination of different models and the integration of a much larger feature set. For reference, the highest MRR scores that were achieved in the associated challenge were 0.689 for the validation set and 0.686 for the confirmation set as opposed to 0.647/0.645 for the benchmark models.

5 CONCLUSION

We presented a new dataset in the context of a recommendation challenge in the online travel domain. We illustrated the characteristics of the data and highlighted some use cases that have already sparked the interest of challenge participants. Initial exploration of algorithmic approaches

to extract information from the data showed that many methods can be applied but also demonstrates that certain aspects of the data such as sparsity and imbalance make it challenging to reach high performance on the evaluation metrics. In addition to the experiments presented here, we will highlight some of the learnings that became apparent in the inspection of multiple models that were developed from different participants of the RecSys Challenge 2019 and demonstrate how they effectively make use of the data.

In the bigger scheme of understanding what methodological approaches are appropriate to deal with the data and predict the items that were clicked, we will use the following section to describe certain aspects that proved successful in the challenge and could be recognized as key ingredients for high-performing results across a variety of teams and submissions, namely feature engineering, gradient boosted models, and ensembling of prediction results.

5.1 Feature Engineering

By far the most successful strategy was to include the extensive engineering of targeted features into the model building process. The participants realized that with a robust set of models and a reasonable validation setup it was hard to overfit on the validation set. As a consequence, the teams built and evaluated a large set of features, many of which made it to the final models. As an example, the winning solution used 220 numerical features that encoded the context of the query and 30 categorical one-hot encoded features [15]. Other submissions in the top 5 had a similar, and sometimes even higher, amount of up to 518 different features [20, 27].

Typical features included item and user features, impression related features, such as absolute and relative prices and characteristics of items that were shown together in a list, as well as session related features. Instead of using sequence aware models that exploit the time-dependent information of session data directly, the participants mostly opted to build sequential features that capture previous interactions and similarity between displayed items and past interactions. Examples for these kinds of features are the time difference between the target click and the previous clickout [15], the number of interactions with a target item in the current session [27], the position of the last interacted item in the impression list [18], or a feature aimed at mimicking the browsing patterns of a user by inferring the expected position of the users' viewport based on previous session interactions [9]. In addition to more conventional features derived from statistics of the dataset, in some cases more complex embedding features were trained to transform the information that is contained in a session into latent variables that could be fed into the prediction models [7, 20].

Most of the feature engineering was done in the context of gradient boosted models. However, features were also engineered for Neural Network models that in other areas provide successful results without the need for extensive crafting of features. For the given data, it was found that even the performance of Recurrent Neural Networks could be improved if they were fed with additional pre-processed data [11].

5.2 Gradient Boosted Models

Almost all teams did not solely rely on the predictions of a single model but followed an ensemble strategy to tweak the performance of their submissions (see Section 5.3). Among the individual models that went into the ensemble, gradient boosted tree methods were preferred. Four, out of the five top-performing teams, strongly relied on the output of gradient boosted models. One reason that the teams relied on these models was that they do not require a careful normalization of the features and allowed for the usage of raw features, which played well with the extensive feature engineering and the inclusion of a large number of estimators that was identified as the key to achieve the best performance on the given dataset.

Table 11. Overview of Performance of Selected Submissions for the RecSys Challenge 2019

Team	Authors	MRR	
		Validation	Confirmation
LogicAI [15]	Jankiewicz et al.	0.689	0.686
Layer 6 AI [27]	Volkovs et al.	0.688	0.685
Meituan Dianping [29]	Wang et al.	0.686	0.684
RosettaAI [18]	Kung-Hsiang et al.	0.682	0.680
TU Dortmund [20]	Ludewig et al.	0.684	0.679
Polytechnic Institute of Viseu [11]	Gama et al.	0.681	0.679
Politecnico di Milano [9]	Damico et al.	0.679	0.677
NVidia [24]	Rabhi et al.	0.673	0.671
KAIST [22]	Oh et al.	0.673	0.670

Algorithm results are sorted in decreasing order of performance.

A popular model choice was the LightGBM implementation of the gradient boosting method that convinced due to the flexibility to handle both classification and ranking objectives, multiple boosting options, and the low memory usage that paid of when training models on the large dataset of the challenge [15, 18, 20, 29]. A strong performance was also achieved with XGBoost models [8] that outperformed Deep Learning models with Transformer and Factorization Machine architectures in direct comparison [18, 27].

5.3 Ensembling

Virtually none of the top teams relied on the performance of a single model for the final predictions. The winning solution consisted of a combination of 37 LightGBM models [15]. Other teams were combining a lower number of models and the exact ensembling strategies differed between teams and depended on the underlying models. A common pattern was the development of different models and the combination of the predictions derived from these models via a linear blend [15, 18, 27] or stacking [9, 28]. Especially for the stacking approach, it was found useful to also include some of the top-performing feature values next to the predictions of the individual models.

Not all teams developed a set of heterogeneous models. Also in the case of similar individual models, aggregating predictions from runs of models that varied in the underlying training data or model parameters proved to be successful. For example, several teams used cross-validation to split the data, calculate predictions on the different folds, and ensemble them for the outcome [19, 22]. Others employed a self-averaging method that combined model outcomes derived with different initialization parameters to reduce the variance in the predictions stemming from Recurrent Neural Networks [11].

Table 11 displays the results for selected submissions that were presented at the workshop on the RecSys Challenge 2019 at the 13th ACM Conference on Recommender Systems in Copenhagen 2019.[7]

We hope that in the future the data can contribute to developing even more and new methods or test already existing ones in a different context and that the exploration of the data will spark the interest of researchers in a variety of areas.

[7] https://recsys.acm.org/recsys19/challenge-workshop/.

REFERENCES

[1] Fabian Abel, András A. Benczúr, Daniel Kohlsdorf, Martha Larson, and Róbert Pálovics. 2016. RecSys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 425–426. DOI: https://doi.org/10.1145/2959100.2959207

[2] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. 2017. RecSys challenge 2017: Offline and online evaluation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'17)*. 372–373. DOI: https://doi.org/10.1145/3109859.3109954

[3] Jens Adamczak, Gerard-Paul Leyson, Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Julia Neidhardt, Wolfgang Wörndl, and Philipp Monreal. 2019. Session-based hotel recommendations: Challenges and future directions. *arXiv:1908.00071*. Retrieved from https://arxiv.org/abs/1908.00071.

[4] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. 2015. RecSys challenge 2015 and the YOOCHOOSE dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'15)*. 357–358. https://dl.acm.org/citation.cfm?id=2798723.

[5] Booking.com. 2019. 515K hotel reviews data in europe. Retrieved July 19 2015 from https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe.

[6] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 527–528. DOI: https://doi.org/10.1145/3240323.3240342

[7] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: The state of the art. *User Model. User-Adapt. Interact.* 25, 2 (2015), 99–154. DOI: https://doi.org/10.1007/s11257-015-9155-5

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. Association for Computing Machinery, New York, NY, 785–794. DOI: https://doi.org/10.1145/2939672.2939785

[9] Edoardo D'Amico, Giovanni Gabbolini, Daniele Montesi, Matteo Moreschini, Federico Parroni, Federico Piccinini, Alberto Rossettini, Alessio Russo Introito, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2019. Leveraging laziness, browsing-pattern aware stacked models for sequential accommodation learning to rank. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359563

[10] Datafiniti's Business Database.2019. Hotel reviews. Retrieved July 19, 2015 from https://data.world/datafiniti/hotel-reviews.

[11] Ricardo Gama and Hugo Fernandes. 2019. An attentive RNN model for session-based and context-aware recommendations: A solution to the recsys challenge 2019. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359757

[12] Goibibo.com. 2019. Indian hotels on Goibibo. Retrieved July 19, 2015 from https://www.kaggle.com/PromptCloudHQ/hotels-on-goibibo.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *Corr* arXiv:1511.06939v4. Retrieved from https://arxiv.org/abs/1511.06939v4.

[14] https://www.spotify.com. 2018. RecSys Challenge 2018. Retrieved August 28, 2019 from https://2018.recsyschallenge.com/index.html.

[15] Pawel Jankiewicz, Liudmyla Kyrashchuk, Pawel Sienkowski, and Magdalena Wojcik. 2019. Boosting algorithms for a session-based, context-aware recommender system in an online travel domain. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359557

[16] 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*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154.

[17] 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 13th ACM Conference on Recommender Systems (RecSys'19)*. 570–571. DOI: https://doi.org/10.1145/3298689.3346974

[18] Huang Kung-Hsiang, Fu Yi-Fu, Lee Yi-Ting, Lee Tzong-Hann, Chan Yao-Chun, Lee Yi-Hui, and Shou-De Lin. 2019. A-HA: A hybrid approach for hotel recommendation. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359560

[19] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Model. User-Adapt. Interact.* 28, 4 (2018), 331–390. DOI: https://doi.org/10.1007/s11257-018-9209-6

[20] 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*. ACM. DOI: https://doi.org/10.1145/3359555.3359561

[21] MakeMyTrip.com. 2019. Indian hotels on Makemytrip. Retrieved July 19, 2015 from https://www.kaggle.com/PromptCloudHQ/hotels-on-makemytrip.

[22] Jaehoon Oh, Sangmook Kim, Se-Young Yun, Seungwoo Choi, and Mun Yong Yi. 2019. A pipelined hybrid recommender system for ranking the items on the display. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359565

[23] Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. 2003. Tree induction vs. logistic regression: A learning-curve analysis. *J. Mach. Learn. Res.* 4 (December 2003), 211–255. DOI: https://doi.org/10.1162/153244304322972694

[24] Sara Rabhi, Wenbo Sun, Julio Perez, Mads Burgdorff Kristensen, Jiwei Liu, and Even Oldridge. 2019. Accelerating recommender system training 15x with RAPIDS. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359564

[25] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current challenges and visions in music recommender systems research. *Int. J. Multimedia Inf. Retriev.* 7, 2 (2018), 95–116. DOI: https://doi.org/10.1007/s13735-018-0154-2

[26] 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, 177–184. DOI: https://doi.org/10.1145/2645710.2645731

[27] Maksims Volkovs, Anson Wong, Zhaoyue Cheng, Felipe Perez, Ilya Stanevich, and Yichao Lu. 2019. Robust contextual models for in-session personalization. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359558

[28] Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. ACM, New York, NY, 783–792. DOI: https://doi.org/10.1145/1835804.1835903

[29] Zhe Wang, Yangbo Gao, Huan Chen, and Yan Peng. 2019. Session-based item recommendation with pairwise features. In *Proceedings of the ACM Recommender Systems Challenge 2019*. ACM. DOI: https://doi.org/10.1145/3359555.3359559

[30] www.xing.com. 2016. RecSys Challenge 2016. Retrieved August 28, 2019 from https://2016.recsyschallenge.com/index.html.

[31] www.xing.com. 2017. RecSys Challenge 2017. Retrieved August 28, 2019 from https://2017.recsyschallenge.com/index.html.

[32] www.yoochoose.com. 2015. RecSys Challenge 2015. Retrieved August 28, 2019 from https://2015.recsyschallenge.com/index.html.

Received November 2019; revised June 2020; accepted July 2020