Robust Contextual Models for In-Session Personalization

Maksims Volkovs
Layer 6 AI
maks@layer6.ai

Anson Wong
Layer 6 AI
anson@layer6.ai

Zhaoyue Cheng
Layer 6 AI
joey@layer6.ai

Felipe Pérez
Layer 6 AI
felipe@layer6.ai

Ilya Stanevich
Layer 6 AI
ilya@layer6.ai

Yichao Lu
Layer 6 AI
yichao@layer6.ai

ABSTRACT

Most online activity happens in the context of a session; to enable better user experience many online platforms aim to dynamically refine their recommendations as sessions progress. A popular approach is to continuously re-rank recommendations based on current session activity and past session logs. This motivates the *2019 ACM RecSys Challenge* organised by Trivago. Using the session log dataset released by Trivago, the challenge aims to benchmark models for in-session re-ranking of hotel recommendations. In this paper we present our approach to this challenge where we first contextualize sessions in a global and local manner, and then train gradient boosting and deep learning models for re-ranking. Our team achieved 2nd place out of over 570 teams, with less than 0.3% relative difference in Mean Reciprocal Rank from the 1st place team. Code for our approach can be found here: https://github.com/layer6ai-labs/RecSys2019

KEYWORDS

Recommender Systems, In-session Personalization, Gradient Boosting, Deep Learning, Transformer, Self-Attention

ACM Reference Format:
Maksims Volkovs, Anson Wong, Zhaoyue Cheng, Felipe Pérez, Ilya Stanevich, and Yichao Lu. 2019. Robust Contextual Models for In-Session Personalization. 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/10.1145/3359555.3359558

1 INTRODUCTION

Many existing recommender systems rely on collaborative filtering (CF) to recommend items to users based on preferences of other users. This approach has proven to be extremely effective in real-world applications, and is deployed in numerous successful online platforms such as YouTube, Amazon and Netflix. However, traditional CF approaches like matrix factorization [5], aim to infer "global" preference patterns across users, and are less effective at capturing local contextual preferences such as those within a session. Session intent can significantly deviate from the global preference profile, for example a user that typically listens to rock can be currently looking for a jazz song. To accurately capture session intent one must take session activity and context into account. As most online consumption happens in a context of a session, accurate and scalable in-session personalization has received increasing attention recently and a number of approaches have been proposed [4, 8, 12].

The *2019 ACM RecSys Challenge* [6] organized by Trivago, focuses on in-session personalization. Given a large dataset of user session logs, the challenge task is to re-rank hotel search results provided by Trivago to accurately capture session intent. Here, we present our approach to this challenge. We conduct extensive feature engineering to capture both global and local context information for each session, and then apply gradient boosting and deep learning models to do the re-ranking.

2 THE CHALLENGE AND DATASET

The organizer of the *2019 RecSys Challenge* is Trivago, a German transnational company that specializes in hotel lodging services. Through its platform, Trivago offers aggregated search results for available hotels in a given area with ability to filter by cost, location, quality etc. In a typical session, a user alternates between searching/filtering for hotels and viewing returned results. Once a suitable option is found, the user then clicks on it (clickout) and gets transferred to an affiliate site to complete the booking. The goal of the challenge is to develop a dynamic session re-ranking model to accurately place clickout item at the top of the returned to the user hotel impression list. Models are evaluated using Mean Reciprocal Rank (MRR):

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

where N is the number of sessions, and rank_i is the rank of the clickout item in session i. The Trivago challenge dataset consists of session logs from November 2018 with 910K training sessions, 291K test sessions with clickout item masked for leaderboard submission and item metadata. Each session is a sequence of one of ten possible actions: six item interactions (clickout, rating, info, image, deals, item search), and four global interactions (sort, filter, destination search, POI search). We partition training sessions forward in time to preserve temporal dynamics, and use 78K sessions for out-of-time model validation. Results on this set closely match leaderboard performance, and we use it for all model validation experiments. A diagram with our data partitioning and challenge re-ranking task is shown in Figure 1.

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

ACM ISBN 978-1-4503-7667-9/19/09…$15.00
https://doi.org/10.1145/3359555.3359558

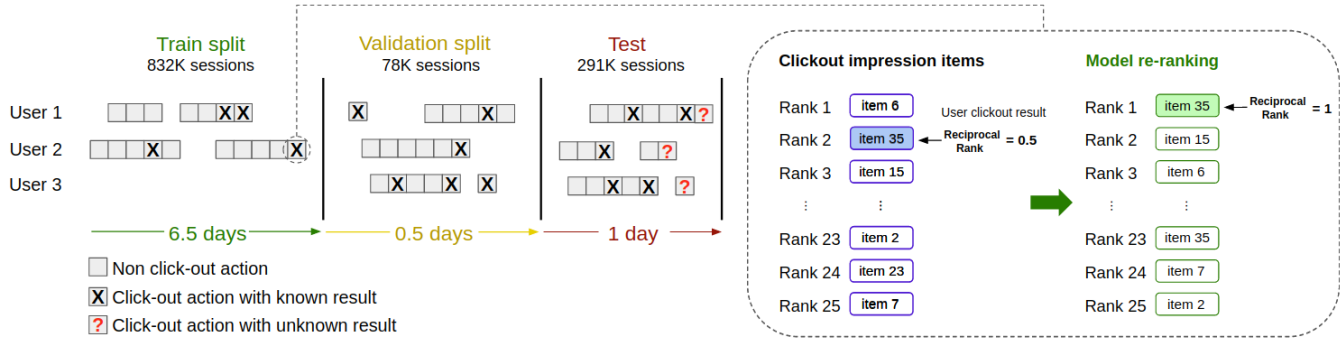

Figure 1: Forward in time data partitioning and challenge re-ranking task. Given a set of ordered impressions (hotel search results) the goal is to re-order them so that the clickout item is at the top of the re-ordered list.

3 OUR APPROACH

Our approach combines gradient boosting and deep learning (DL) models trained on item features that are extensively engineered to capture both session context and item properties. To train the models we create training instances by extracting features for all impression items in the *last* clickout of every session. For every item we extract 330 features and assign a binary label of 1 if it is a clickout and 0 otherwise. This gives a training set of 14.5M instances and a validation set of 1.8M instances. The features roughly divide into four categories and describe session context activity, item statistics within the session, item statistics across sessions and statistics for all items in current impression. Below we provide some examples of important features in each category:

Session Context

- Item actions that precede clickout
- Non-item actions that precede clickout
- Number of session steps
- Time duration of session
- Device used

Item Session Statistics

- Price and price rank in impression
- Rank position in impression
- Previous actions on this item in current session
- Item dwell time in current session
- Item properties metadata

Item Global Statistics

- Global item action counts across sessions
- Impression statistics across sessions such as average position, price and price rank
- Number of unique users that interacted with this item
- Difference between current session and global statistics

Impression Items Statistics

- Price summary across impression items
- Metadata properties across impression items
- Item similarity within impression
- Global statistics summary across impression items

Throughout feature engineering we consistently found that performance can be significantly improved by combining both local session features and global features across sessions. Global features are particularly useful for short sessions where little context information is available. We also found it beneficial to extract features that summarize all items in the current impression, and contrast them with the target item. Users tend to make contextual choices often selecting the cheapest or best reviewed hotel from the shown list. Summarizing impression items enables the model to learn this dynamic and improves performance.

3.1 Gradient Boosting

Our strongest model is a gradient boosted tree ensemble (GBM). We use an excellent XGBoost [1] library and conduct extensive experiments to maximise validation performance. Through these experiments we find that the most important training hyper-parameters are tree depth, learning rate, feature sampling rate, and L2 regularization. Major advantage of gradient boosting is that virtually no feature normalisation is required. We were able to successfully train on the raw feature set that included categorical, ordinal and numeric features. This is not possible with deep learning models, and we spent considerable effort on feature normalisation to make gradient optimization work (see following section).

We train GBM with binary cross entropy objective to predict clickout probability for every item in impression. To take advantage of item impression grouping, we experimented with ranking and softmax objectives that take into account the entire impression list. However, independent binary cross entropy consistently performed better. During inference, we make a forward pass through the GBM and re-rank impression items according to the probability of clickout.

3.2 Deep Learning

Advances in attention mechanisms and in particular the transformer architecture [9], have resulted in substantial accuracy gains on many sequence-based tasks [2, 10]. Recently, a number of attention and recurrent architectures have been proposed for session modeling in recommender systems [7, 8, 12]. Inspired by this work we explore attention-based architectures to jointly model items in each impression list. The best architecture that we found consists of three stages: (1) input embedding, (2) transformer blocks and (3) feed forward layer. A diagram of this model is shown in Figure 2, and we give a brief overview of each stage below.

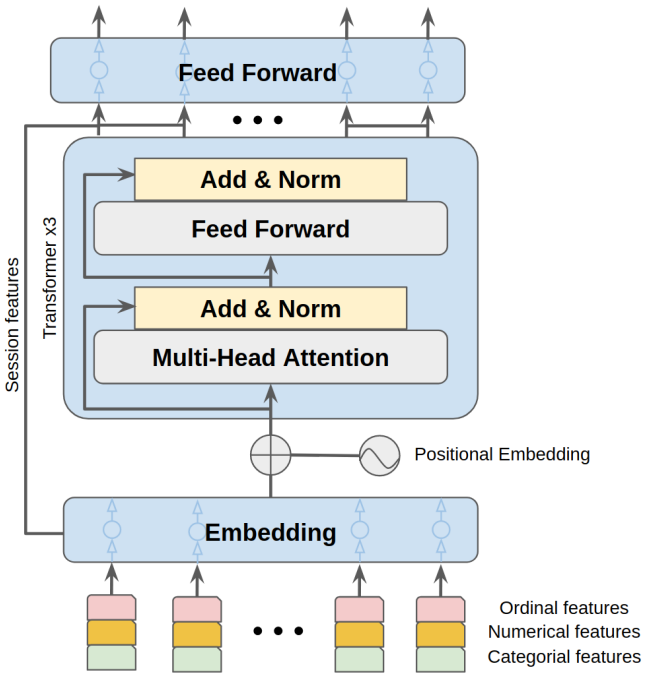

Figure 2: Our three-stage transformer architecture.

Input Embedding. The goal of this stage is to transform input features into a representation that is suitable for gradient optimization. Given that features contain ordinal, categorical and numeric values, we found that appropriately representing each type has significant effect on model performance. For dense features we use the Yeo-Johnson transform (see Equation 2 in [11]) where the parameter λ is chosen separately for every feature to minimize skewness. Categorical features are represented with learned embedding vectors with dictionary size equal to the number of categories. The embedding dimension is set through cross validation for each feature. Finally, we also use embeddings for ordinal features with the exception that negative and positive values are represented with the same embedding: $x \mapsto \text{sign}(x) * \text{embed}(|x|)$. Negative ordinal features arise when we consider position and price rank differences between items. Multiplying an embedding by sign enables to simultaneously model both positive and negative differences without doubling the dictionary size.

Even with all of the above transformations, we found it difficult to match GBM performance with deep learning models. To further enhance our input representation, we experimented with various ways of extracting information learned by GBM and encode it as additional input representation. Previous work has explored using prediction leaf indices from a trained GBM tree model as input representation to other models [3]. We found this approach to also work well here. Given a trained GBM model with T trees, we extract prediction leaf indices for every instance by passing it through the model. This results in a highly sparse representation with exactly T non-zero values. We then compress this representation by applying randomized truncated SVD to get a dense embedding for every instance. These embeddings summarize information that GBM encodes with its tree-based structure. A typical configuration that has worked well here is a 1000 tree GBM model with depth 10, which results in ~2M-dimensional leaf index instance representation that we then compress to 128 dimensions.

Table 1: Final leaderboard with top 5 teams, our team "Layer 6 AI" is shown in bold. We also show our best single model which would rank 4'th on the leaderboard.

Rank	Team name	MRR (final)	ΔMRR from 1st place
1	LogicAI	0.685711	
2	**Layer 6 AI**	**0.685497**	**0.03%**
3	pvz	0.684071	0.24%
4	RosettaAI	0.679933	0.84%
5	letoh govatri	0.679299	0.94%
	single model	0.682834	0.67%

Collectively these feature transformations improve MRR performance by over three points for most DL architectures that we experimented with when compared to raw features. After input processing we separate the features into session-based and item-based subsets. Item-based set is then passed through the transformer encoder and concatenated with session-based set before feed forward layer.

Transformer Blocks. We apply standard multi-head transformer blocks proposed by [9]. Repeated self-attention enables the model to focus on different parts of the impression item sequence, and collectively make a decision on which item will result in clickout. To preserve positional information, which is important here as users typically scan impressions from top to bottom, we add positional embeddings to item representations.

Feed Forward Layer. Representations from the transformer encoder are concatenated with session-based features, and passed to the feed forward layer that outputs clickout probability for each item. Weights are shared between items so feed forward layer is structured like a 1D convolution with stride 1, and operates along the item impression list from top to bottom. Clickout probability is obtained by applying sigmoid function to feed forward output for every item, and we use binary cross entropy loss to optimize the model:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{K_i}\sum_{j=1}^{K_i} y_{ij}\log\hat{y}_{ij} + (1-y_{ij})\log(1-\hat{y}_{ij}) \qquad (2)$$

where N is the number of sessions, K_i is the number of impression items in session i, y_{ij} is 1 if item in position j is clickout and 0 otherwise and $\hat{y}_{ij}$ is the predicted clickout probability.

4 EXPERIMENTS

All experiments were conducted on Ubuntu machines with Intel Xeon CPU E5-2620 v4 @2.10GHz, 256GB RAM and Nvidia Titan V GPU. All models were tuned using the validation set (see Section 2), and then submitted to the evaluation server. We generally observed that validation performance translated well to leaderboard with a consistent MRR difference of ~ 0.08 between the two scores.

GBM. After tuning the XGBoost hyper parameters to maximize validation performance, we found the following settings to be important: `eta = 0.1`, `max_depth = 10`, `colsample_bynode = 0.8`, `lambda = 4000` and `alpha = 10`. Validation results for this model are shown in Table 3. From the table we see that GBM is the best

Table 2: Top 20 most important features for the GBM model.

Feature	Gain (%)
Is target item in top position in impression (boolean)	42.7
Target item position in impression	9.8
Target item star rating ranking in impression	6.1
Price rank within items above target item in impression	4.2
Number of actions on target item in current session	3.3
Global average price rank for target item position in impression	3.1
Item-based neighbor similarity between user and target item	3.0
Average price rank of items with same star rating in impression	2.6
Difference in price rank between target item and last item that user interacted with before clickout	2.2
Price rank within items with same star rating in impression	1.7
Impression position of last item that user interacted with before clickout	1.5
Difference in position between target item and last item that user interacted with before clickout	0.9
Number of items with the same star rating in impression	0.8
Impression length	0.7
Number of actions between clickout and last item action	0.5
Time between clickout and last item action	0.4
Number of items with same user rating in impression	0.4
Number of previous impression lists in session that exactly match current impression list	0.4
Global same length impression list count	0.4
Target item price rank in impression	0.4

performing model, and DL models even with SVD leaf embeddings are not able to match it. Table 1 shows final leaderboard results. Our best submission was a blend of GBM and DL models, but we also show results for our best single GBM model. We see that our single model would have achieved 4'th place with less than 0.67% relative difference in MRR from the top team. Blended models are challenging to deploy and maintain in production, and these results indicate that even with a single model we can get highly competitive performance.

To gain additional insight into this problem we analyze feature importances for the best performing GBM model. Table 2 shows the top 20 most important features together with their relative importance scores. The importance scores are computed by averaging loss reduction across all node splits for each feature, and then normalizing across features. From the table we see that top 10 and top 20 features cover 78.7% and 85.1% of the importance respectively.

We observe several patterns from the types of features that our model places high importance on. First, ordinal features such as position in impression, rank by price, star and user rating consistently appear at the top of the table. This indicates the importance of factors like price and rating in hotel selection. Moreover, as we discussed above, users tend to make contextual choices and often select cheapest or highest rated hotel for the list. We consistently found that ordinal features such as relative rank within impression, worked significantly better than absolute ones. Second, as expected, session context plays an important role in the clickout decision. Features like actions right before clickout and actions on target item in current session, frequently appeared in many of the top feature lists that we analyzed. These features allow the model to better capture session intent and re-rank the items accordingly.

Table 3: Validation model performance.

Model	Validation MRR
LSTM	0.6721
LSTM+SVD	0.6750
Transformer	0.6734
Transformer+SVD	0.6765
GBM	0.6771
Blend	**0.6798**

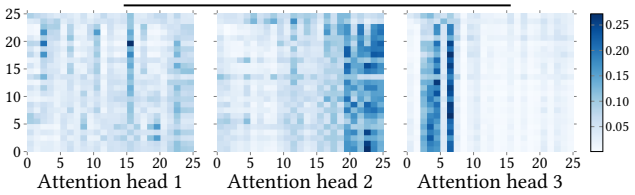

Figure 3: Attention head scores for a validation session.

Finally, global statistics across sessions also have high importance. Global statistics further enable the model to contrast current session against a "typical" global session and act on deviations between the two.

Deep Learning. All deep learning models were trained until convergence with Adam optimizer and batch size between 50 and 100. Most optimization runs require 70K-100K updates with a wall-time of at most 18 hours for the longest run. In addition to the transformer architecture outlined in Section 3.2, we also experiment with LSTM models. For transformer model we use three attention blocks with three attention heads in each block, and hidden dimensionality of 75. LSTM models have bi-directional architecture with state size set to 100. Validation MRR results are shown in Table 3. From the table we see that our transformer architecture generally outperforms LSTMs, but the gains are relatively small. We also see that adding leaf index SVD embeddings (+SVD) consistently improves performance for both models. This indicates that even our elaborate input pre-processing pipeline is not sufficient. A promising future direction is thus to further investigate feature extraction techniques suitable for DL models with emphasis on complex categorical and ordinal features. Linearly blending LSTM, transformer and GBM models gives a further boost in MRR as shown by the "Blend" model in Table 3. This is our best performing submission that achieved 2nd place on the final leaderboard.

Figure 3 shows self attention scores for each of the three attention heads in the first transformer block for one of the validation sessions. We see that each head specialises on different regions in the impression list. First head generally spans the entire list versus second and third heads concentrate on items in bottom and top positions respectively. This is one of the main advantages of attention-based architectures, as manually extracting analogous features for different regions of the impression list is an extremely time consuming and laborious task.

5 CONCLUSION

In this paper we outlined our approach for the *2019 RecSys Challenge*. We conducted extensive feature engineering to describe both within and across session user behavior. We then trained a combination of gradient boosting and deep learning models to achieve 2nd place on the final challenge leaderboard.

REFERENCES

[1] 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)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (ADKDD'14)*. ACM, New York, NY, USA, Article 5, 9 pages. https://doi.org/10.1145/2648584.2648589

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[5] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[6] 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)*. ACM, New York, NY, USA, 2. https://doi.org/10.1145/3298689.3346974

[7] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling user session and intent with an attention-based encoder-decoder architecture. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 147–151.

[8] 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*. ACM, 130–137.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. 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., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[10] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).

[11] I.K. Yeo and R.A. Johnson. 2000. A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika* 87, 4 (2000), 954–959. http://www.jstor.org/stable/2673623

[12] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention networks. In *the 27th International Joint Conference on Artificial Intelligence*.