Context-aware Session-based Recommendation

An Attentional Deep Learning Approach to Re-ranking

Tomás Alexandre de Menezes Pereira
tomas.m.pereira@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
October 2021

***Abstract*—Recommender systems mediate a substantial share of the online interaction interface, providing an ever-more relevant array of benefits for both users and business providers amidst the exponential growth of available digital content, products and services. Some domains, such as travel and e-commerce applications experience sporadic and mostly anonymous activity, thereby only observing short-term session-based dynamics, requiring robust solutions independent of structured user profiles. This work encompasses the development of an attentional deep learning re-ranking recommender under these challenging conditions, on the basis of the dataset provided for the 2019 ACL Recommender Systems Challenge. The developed approach demonstrates an 85th percentile result (in predicted online MRR, considering a limited score range exceeding that of the provided baseline) with minimal feature generation effort in a very restricted computational environment. Nonetheless, various inconsistencies in the research field including the prominent divergence between offline and online production objectives, motivated a shift in the main focus away from leaderboard performance, which typically rewards feature over-engineering and model complexity. An analysis of representation learning's potential and specific component impact, derived from an automatic Bayesian Optimization with Gaussian Processes procedure over a highly conditional hyperparameter space, was emphasized instead.**

***Index Terms*—Recommender systems, session-based recommendation, sequential interaction modeling, deep learning, self-attention, bayesian optimization**

I. Introduction

Recommendation methods have been refined over time to further enhance the personalized user experience, usually only made possible with detailed knowledge and deep understanding of the item space, the users, their preferences and behavior, and additional contextualization. Item similarity and neighborhood-based methods have given way to latent factor models that take different users' interaction information into account and hybrid strategies that can better tackle cold start situations and provide multi-scale data modeling. The generalization of these concepts with neural network-based architectures, for instance, has allowed for increased control over the incorporation of crucial multi-modal contextual signals, enabling circumstantial factors to adapt the recommendation space, among an array of other advantages [1]. This is especially useful in the sequential recommendation setting, where temporal patterns and trends from past implicit and explicit feedback, usually derived from activity logs, can be complemented with features modeling the continually changing needs and interests of users to infer their intent and predict future interactions.

Nonetheless, some domains such as travel and e-commerce applications experience sporadic and mostly anonymous activity, thereby only observing short-term dynamics, requiring alternative solutions independent of structured user profiles. This work encompasses the development of a recommender under these challenging conditions on the basis of the dataset provided for the 2019 ACL Recommender Systems (RecSys) Challenge, using state-of-the-art methods such as attention mechanisms, ubiquitous in modern approaches. The misalignment between most offline objectives and those relevant in production environments motivated a shift in focus away from maximum leaderboard performance achievement with traditional competition strategies such as the usage of overly complex models or hundreds of input features. Instead, the model's architectural optimization process, usually disregarded in the literature, was emphasized in an exploration of representation learning's potential. Additionally, the developed model was tested against simpler rule-based baselines which have recently produced competitive results even against the state of the art in sequential recommendation, and its specific component impact was also assessed.

Subsequently, this work's research objectives (ROs) were reduced to three main points:

- RO.1 Development of a competitive modular deep learning re-ranking recommender system for sparse session-based domains, focused on representation learning;
- RO.2 Application of state-of-the-art processing techniques and methods, such as self-attention mechanisms to help capture user preferences and intent from behavioral interaction sequences;
- RO.3 Implementation of an automatic bayesian optimization process for the model's architecture, subject to a highly conditional hyperparameter space.

The next Section **II** introduces the RecSys 2019 Challenge. Section **III** provides an overview of session-based literature and domain challenges. The first methodology Section **IV** outlines the problem formulation. It is followed by descriptions of the data and feature processing in Section **V**, and the model's optimization process in Section **VI**. The results are presented in Section **VII**. Finally, some brief conclusion points are provided in Section **VIII**.

II. The RecSys 2019 challenge

In its 2019 edition, the annual ACL RecSys Challenge was organized by *trivago*, TU Wien, Politecnico di Bari and Karlsruhe Institute of Technology with the goal of exploring context-aware recommendation on a highly sparse session-based setting, unsuitable for most traditional approaches [2].

The selected digital travel domain is subject to a variety of obstacles, including the constantly changing prices, offers, and availability of a vast collection of accommodation, the extreme cold-start induced by the infrequent listing browsing by mostly anonymous users, and the highly dynamic search criteria motivated by their variable long and short-term preferences and trip-specific needs [3].

Fig. 1. *trivago*'s website interaction distribution with the user action types available in the RecSys19 dataset presented in Table II, adapted from [3].

trivago's global search platform aggregates localized accommodation information (including price, rating, visual and textual descriptions, among others) based on user queries. Users can interact with the listings in multiple ways, some of which are presented in Figure 1, and click on relevant ones to be redirected to external affiliate booking sites where transactions can be completed.

Naturally influenced by the platform's primary revenue stream centered on cost-per-click [4], the technical objective of the challenge consisted in the prediction of which items were more likely to be clicked at the end of each of their user's largest sessions, from lists of items (impressions) presented at these events, to improve the ranking process. This required the re-ordering of the impression lists by click likelihood (representing contextual relevance), which was evaluated with the Mean Reciprocal Rank (MRR),

$$\text{MRR} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{\text{rank}_j}, \tag{1}$$

where N is the number of evaluated lists (samples) and rank_n is the predicted position rank for the clicked item (target label) in sample j. The Reciprocal Rank's value, and consequent new ranking performance for a sample, increases with the proximity of the target to the top of each list, reaching its maximum of one for ranked lists where the clicked item appears in the top position ($\text{rank}_j = 1$).

The data provided by *trivago* was comprised of anonymized user session interaction logs recorded in their platform over eight days, 01-08 November 2018, and an additional accommodation metadata database with item-specific attribute lists. Each log entry is characterized by the features displayed in Table I, consisting of a single user interaction over a given item, from the fixed set of ten available types presented in Table II.

The logs were provided in separate training and test sets, with the latter consisting of events occurring in the last two days, for which the ground truth labels of the clicked items in the largest user sessions were omitted. Up until the challenge's deadline of July 2019, the user-submitted entries, consisting of predicted rank lists for the relevant click event items, were evaluated on an online platform which has, since then, ceased to operate. At the time of writing, the omitted test labels have not yet been made publicly available for offline use. As such, the data regarded in this work is limited to the six-day training set, with its almost 16 million actions, 730 803 users, and 910 683 sessions, from here on defined as RSC19. Hence, to obtain comparable results, the data was split according to the process applied by the 7th overall placed team *Mustelideos* [5], which considered every click a training sample, and reproduced a setting similar to that found in the original challenge's test dataset, omitting the labels for each user's last largest session clicks in the final two training days (05 and 06 November). For hyperparameter optimization purposes, a subset of the resulting training data was split into smaller validation and test sets, corresponding to the last largest session clicks in the third and fourth days, respectively.

Table I. RSC19 original features.

Original feature	Description
User ID	User identifier
Session ID	Session identifier
Timestamp	UNIX timestamp in seconds for the time of interaction
Step	Time step in the sequence of events within the session
Action	Action type for each event (see Table II)
Reference	Reference identifier for each action (see Table II)
Platform	Country of the web platform used for the search
City	City name for the session search context
Device	Device used for the search
Filters	List of active filters at a given timestamp
Impressions	List of item identifiers displayed to the user in a click event
Prices	Corresponding list of nightly prices for the impression items
Metadata	Specific item attributes (features and amenities)

Table II. Types of user actions and respective action references.

Action type	Description	Reference
Clickout	Item click that redirects the user to an affiliate website	Item ID
Interaction item rating	Interaction with an item's rating/review elements	Item ID
Interaction item info	Interaction with an item's information elements	Item ID
Interaction item image	Interaction with an item's images	Item ID
Interaction item deals	Interaction to extend a given item's affiliate price deals element	Item ID
Search for item	Specific accommodation search	Item ID
Change of sort order	User determined sort of the presented impressions, by price, distance, rating and popularity	Sort type
Filter selection	User determined impression filtering by feature (e.g., amenities or minimum number of stars)	Filter type
Search for destination	Specific location-based filtering	Location
Search for POI	Specific point of interest-based filtering	Point of interest

III. Related work

With the ability to model complex non-linear user-item interaction patterns and with the capacity to learn joint representations of multi-modal structured and unstructured data [6], allowing for the incorporation of various contextual signals, deep learning architectures have greatly influenced the recommender system space, dominating the state-of-the-art in recent years [1]. The neural generalization of traditional methods, such as matrix factorization, expanded onto the sequential field, with most new solutions generating ranked predictions of top-K items most likely to be interacted with in the near future or after a given action, as shown in Figure 2.

Sparser session-based settings, such as the one studied in this work, where individual user information is limited or

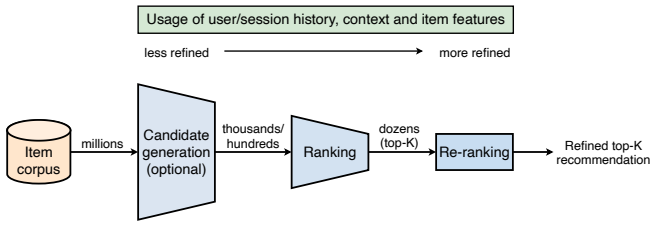

Fig. 2. Modern recommendation framework, modified from [7].

completely unavailable and cold-start scenarios are frequent[1], have been shown to benefit significantly from short-term intent representations derived from implicit sequential interaction patterns. Due to its unmatched sequence processing ability in a variety of tasks, the use of Recurrent Neural Network (RNN)-based architectures in this domain is naturally ubiquitous.

Hidasi et al. [8] created the foundation for top-K next-item session-based neural recommenders by introducing a second order RNN architecture with negative output sampling to manage the large output space in the RSC15 clickstream dataset, GRU4Rec. Gated Recurrent Units (GRUs) were found to outperform the more complex Long Short-Term Memory (LSTM) units with numerous one-hot encoded item ID input sequence variations. The specialized pairwise ranking loss functions introduced in their work were later found to promote vanishing gradients and were modified in [9], which also showed a comparatively good performance from an altered categorical cross-entropy log loss. Tan et al. [10] further improved the GRU-based model with embedding dropout, an adaptation to temporal changes and data processing techniques. A direct embedding output similar to that of [7]'s in candidate generation was proposed with cosine similarity instead but a high-dimensional softmax cross-entropy approach was still found to perform better. [11] developed a hierarchical RNN model, in which a top-level predicts initializations for a lower-level GRU modeling the sequential session information. If user identifiers are available, the top-level can easily relay the evolution of each user's interests over time and across sessions.

The incorporation of additional context features besides past item ID interaction sequences has also shown to be crucial in these domains. [12] developed an architecture with three parallel GRU-based structures to process item IDs with descriptive image and text feature vectors simultaneously. Smirnova and Vasile [13] proposed a conditional RNN model and considered supplemental temporal and event type features, with different stage context injection at the input, unit dynamic and output structural levels.

As in sequence-based settings, works including [14, 15] have consolidated the importance of session-based attention. Because interest drifts that lead to redundant clicks are relevant in shorter timespans, [14]'s hybrid attentional encoders (with global and local focus) emphasize the user's main session purpose in item click sequences, producing a unified session representation used to compute click probabilities for available items, trained with a categorical cross-entropy loss. Liu et al. [15] built upon this solution, introducing higher attention dependency with a memory priority model, more capable of deriving user intent from smaller sessions. The use of transformers has started to become equally central in these domains, with [16], for instance, outperforming previous non-attentional recurrent approaches in next-item recommendation.

[1]Greatly limiting the performance of user-dependent neighborhood and matrix factorization methods.

Challenges and Limitations

Most modern recommendation approaches, including the ones mentioned earlier in this section, are concerned with the production of ranked item lists from the entirety or majority of the item corpus[2]. This objective differs from that of the challenge studied in this work, resulting in the inability to directly compare the presented methods to the developed solution. The item list conditioning experienced by the re-ranking process can be easily lifted, and adaptations to the high-dimensional output space, using hierarchical softmax, negative sampling, or output embedding generation with nearest neighbors, for instance, can be quickly implemented. However, unlike in the prevalently studied e-commerce domain, where inter-category product exploration can occur in an almost frictionless manner, the travel domain's extreme location dependence combined with the lack of item-specific location metadata in the RSC19 dataset prevented full corpus-based tests.[3]

However, problems in recommender systems research extend far beyond these unaligned objectives. The large variety of public and private datasets with endless variations, evaluation metrics, validation procedures, lack of optimization details for reproducibility, and baselines used in the field has left researchers struggling to measure progress, questioning state-of-the-art advances. Recent works have found that the lack of standardization and solid benchmarking has led to most datasets only being equally considered in a very small amount of relevant papers [17]. Metrics defining the performance of developed algorithms can range from the traditional information retrieval Precision and Recall to the Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), Click Through Rate (CTR), and the previously introduced MRR, among others [17, 18]. These, matched with another assortment of validation processes, are often chosen based on past usage without additional justification, usually with also arbitrary list cutoff sizes. Most importantly, maximizing offline user-centric objectives such as CTR might not even reflect the intended business-centric purposes of the system (e.g., additional revenue) [18]. Some competition-based advances are ignored due to the unjustifiable engineering effort needed for their production environment application. YouTube, for instance, preferred a general unified deep learning pointwise system over more complicated ensemble or better offline performing systems for efficiency and scalability reasons [7].

Deep learning applications are also affected by these drawbacks and the subfield's lack of direction is concerning. [17, 19, 20] found that "computationally and conceptually" simpler alternatives including association rule, nearest-neighbor, and even popularity aggregator methods could outperform algorithms such as NCF [21] or base GRU4Rec [8][4].

[2]Corresponding to the first ranking stage of Figure 2.

[3]In most cases, it would not make sense for hotels in Dubai to show up in a London-based search session. In RSC19, the same items can appear associated to dozens of different search cities, complicating location-based grouping.

[4]Although in offline conditions more comparable to data science competitions where popularity bias, small corpus coverage and scalability issues inherent to most can be disregarded.

The inherent unpredictability of user behaviors, combined with the lack of intent ground truth support in implicit signals, training feedback loops and an array of biases, such as the presentation bias clearly visible in RSC19's click distribution by impression position in Figure 3, further increase the need for robust strategies.

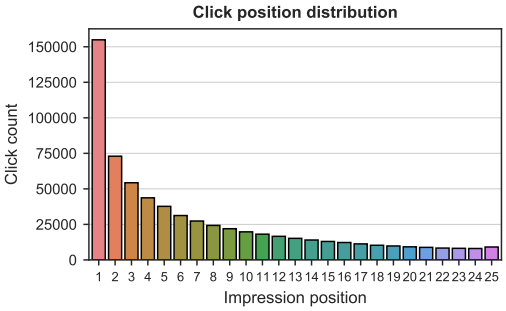

Fig. 3. Imbalanced clicked item position distribution. Lower position indices are equivalent to higher layout position.

IV. Problem Formulation

The re-ranking of the accommodation impression lists required by the challenge can be framed as a supervised learning, multiclass classification task solving the proposed implicit click prediction surrogate problem. To predict the clicked item's position at any given click event in a user session y, a deep learning model was designed to generate an array of click probabilities $\hat{\boldsymbol{y}}$ for the displayed impression list items, leveraging previous session-based sequential behavior signals and numerous types of additional context as input. These probabilities are used to calculate the negative log likelihood cost (NLL), according to which the model is trained, with the maximum value's index contributing to the classification evaluation. The ranking performance is given by the MRR of the sorted probability indices, $\boldsymbol{y}_{\text{rank}}$, as shown in the high-level modeling process for a session sample input displayed in Figure 4.

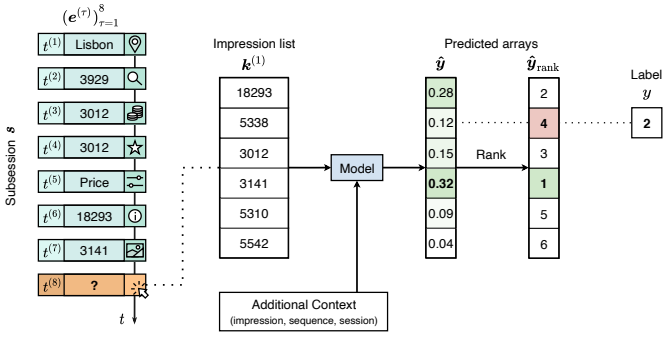

Fig. 4. High-level simplified modeling process for the eight event session sample (subsession) $\boldsymbol{s}$. The model outputs a vector $\hat{\boldsymbol{y}}$ with the click likelihood for each of the impression list $\boldsymbol{k}^{(1)}$'s item positions. A generalized transformed context input is represented, consisting of dynamic and static features captured up to the click event's timestamp $t^{(8)}$ (unknown at prediction time). The click probability vector is sorted to produce the position rank vector $\hat{\boldsymbol{y}}_{\text{rank}}$. Since the predicted rank for the ground truth label y is 4, the RR for this example is 1/4.

Formally, let $\boldsymbol{S} = (\boldsymbol{e}^{(1)}, ..., \boldsymbol{e}^{(n)}) \equiv (\boldsymbol{e}^{(\tau)})_{\tau=1}^{n}$ denote an arbitrary user session with n sequential events. Each of these events consists of an interaction a, from the ten possible types previously introduced in Table II, with a reference item i at time t contextualized[5] by $\boldsymbol{c}$, such that the τ^{th} event can be represented as the tuple $\boldsymbol{e}^{(\tau)} = (a^{(\tau)}, i^{(\tau)}, t^{(\tau)}, \boldsymbol{c}^{(\tau)})$. In particular, a single session contains a subsequence of $w \in \{1, ..., n\}$ click events, $(\boldsymbol{e}_h^{(\tau_h)} | a^{(\tau_h)} = \text{clickout})_{h=1}^{w}$, with $\boldsymbol{e}^{(n)} = \boldsymbol{e}_w^{(\tau_w)}$, i.e., the last session event corresponds to a click. Click events are crucial to the problem as they contain the impression lists presented to the users, $\boldsymbol{k}^{(h)}$ (for every arbitrary $\boldsymbol{e}_h^{(\tau_h)}$), which in turn hold the targets. Based on these, the model was trained to produce a sequential prediction for what item in the impression list is clicked at $t^{(\tau_h)} + \epsilon$, where ϵ is a small time interval, by calculating click probabilities for each one, $\hat{\boldsymbol{y}}^{(h)} = P(y^{(h)} = k_p^{(h)} | \boldsymbol{x}^{(\tau_h)}, \boldsymbol{\theta})_{p=1,...,|\boldsymbol{k}^{(h)}|}$, as noted above.

To leverage every click's information, the training set was augmented with a vectorized implementation of the method used in [10, 14, 15], such that every session produced $w - 1$ additional subsessions (training samples) containing the events preceding each of the w session click events, $\boldsymbol{s}^{(l)} = (\boldsymbol{e}^{(\tau)})_{\tau=1}^{\tau_l}$, represented below in Figure 5.

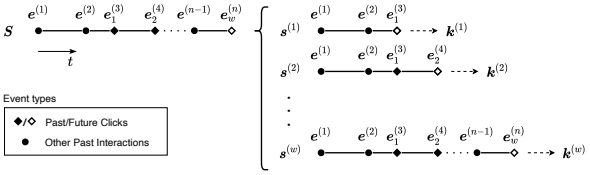

Fig. 5. Augmentation process for session $\boldsymbol{S}$, which generates $w - 1$ new subsessions $\{\boldsymbol{s}^{(1)}, ..., \boldsymbol{s}^{(w-1)}\}$, $(\boldsymbol{s}^{(w)} = \boldsymbol{S})$, based on the w click events, each associated with the corresponding impression list $\boldsymbol{k}^{(h)}$.

The process described above was repeated for every session $\boldsymbol{S}_{\forall j \in N}^{(j)}$ in the dataset $\mathcal{D}$, for which the session axis had been omitted in simplification.

V. Data Processing

This section encompasses the rationale behind the development of the recommender's input $\boldsymbol{X}$, consisting of relevant features generated and extracted from the raw activity log dataset, such that the contextual signals and patterns depicting user intent are captured efficiently in complete representations.[6]

A. Preprocessing

The initial preprocessing stage consisted of general data structural modification, cleansing and filtering steps, mostly based on dataset exploration and visualization, as preparation for subsequent feature-based operations.

Duplicate log entries, interactions with undefined reference values, invalid click events with impression lists missing the clicked reference and subsessions without clicks were removed. Additionally, user sort interactions were incorporated as filter selections.

The original dataset's event distribution was dominated by image interactions, which accounted for almost three quarters of the events largely due to similar consecutive actions with the same reference items. So as to not overpower the interaction

[5]For simplicity, $\boldsymbol{c}$ corresponds to a vectorial proxy for different time-dependent and independent contextual signals.

[6]General processing was done in `Python 3.7.7`, on a 6-core 2.6GHz i7, 16GB RAM machine. Model implementation was done in `TensorFlow 2.3.0` and training was executed on Google Colab's GPU mode subject to the free dynamic usage limits, in https://research.google.com/colaboratory/faq.html.

sequences, these consecutive actions were grouped into single events, returning the respective temporal endpoints and generating a frequency feature, corresponding to each group's size, majorly reducing the action distribution imbalance. [7]

Instead of removing only sequences of length one as in most of the literature, sessions and initial subsessions with less than three events were dropped. This was done to ensure model robustness by reducing unwanted stochasticity from training due to low contextualization, as it was expectedly noted that the first session time steps mainly consisted of more exploratory, non-item specific interactions.[8]

In the end, the original 910 683 user sessions were reduced by approximately 36% down to 332 849, containing 660 526 samples and 2 995 184 events with more even action time and count distributions. There was an average of two clicks per processed session, whose size remained relatively small, with only 15% containing more than ten time steps, although their median event number increased from four to six.

B. Feature engineering and representation

The feature set was divided into three main different categories and two additional subcategories, expanded in Table Table III, according to their characteristics and behavioral structure across training examples:

- Interaction sequence features, $\boldsymbol{X}_{\text{seq}} \in \mathbb{R}^{n_{\text{seq}} \times m_{\text{seq}}}$, model dynamic sequential information across a session leading up to a given click event;
- Session features, $\boldsymbol{X}_{\text{ses}} \in \mathbb{R}^{m_{\text{ses}}}$, consist of static context signals that define each subsession sample;
  - Filter features, $\boldsymbol{X}_{\text{filt}} \in \mathbb{R}^{d_{\text{filt}}}$, comprise subsession-based search and sort filters selected by the users;
- Impression features, $\boldsymbol{X}_{\text{imp}} \in \mathbb{R}^{n_{\text{imp}} \times m_{\text{imp}}}$, describe and summarize properties and interactions for the impression list's items at every click;
  - Metadata features, $\boldsymbol{X}_{\text{meta}} \in \mathbb{R}^{n_{\text{imp}} \times d_{\text{meta}}}$, contain additional impression characteristics retrieved from the metadata database.

Table III. Feature space. *Original features, $^{+}$Vectorized inter and in-session time accumulators, $^{\blacklozenge}$Accompanied by boolean features for value existence. Time dif. is the time delta between general interactions and actions of a given type. Dwell time is given by the interaction time plus the time delta up to the following interaction.

Sequential ($\boldsymbol{X}_{\text{seq}}$)	Session ($\boldsymbol{X}_{\text{ses}}$)	Impressions ($\boldsymbol{X}_{\text{imp}}$)
• Ref. item ID*	• Subsession	• Imp. item ID*
• Action ID*	• Total substeps	• Position
• Step*	• Total time steps	• Price*
• Frequency	• Total session time	• Views^{+}
• Session time$^{\blacklozenge}$	• Imp. list length	• Clicks^{+}
• Time dif.$^{\blacklozenge}$	• City ID*	• Interactions^{+}
• Dwell time$^{\blacklozenge}$	• Platform*	• Dwell time^{+}
	• Device*	• Metadata ($\boldsymbol{X}_{\text{meta}}$)*
	• Filters ($\boldsymbol{X}_{\text{filt}}$)*	

The processed dataset had to then be transformed in order to create suitable numeric representations for the deep learning model.

For nominal categorical features, an approach similar to that of [7] was taken, with those belonging to the same vocabulary/ID space sharing the same embedding layers but being separately input to the network, such that specialized representations are joint-learned per feature by deeper layers, with added efficiency and generalization benefits. In this case, four ID spaces were considered: the item ID space, shared by reference and impression item IDs, the attribute ID space, shared by filters and item attributes, and the remaining individual action ID and city ID spaces. These vocabularies were created after a full data pass[9] and consisted of lookup tables mapping the original IDs into sequential integers, which are then subsequently mapped to the corresponding vector embeddings. Out-of-vocabulary values, such as non-accommodation reference IDs, were mapped to the 1 integer token.

Regarding continuous and ordinal features, the Min-max normalization method is typically used to linearly rescale each variable to the [0,1] range. However, unbounded counter or time-based features are especially prone to legitimate extreme observations, sometimes highly influential in the distributions, which might contain valuable information that would be wrongly discarded if artificial value limits were to be imposed [22]. Therefore, Quant, the non-linear method applied in [7, 23], consisting of a uniform distribution mapping from an estimate of the feature's cumulative distribution function, was implemented instead yielding better results. Statistics were computed on the training set only, as to prevent information leakage from the test data.

Sequences were post-padded with zeros, in the case of embedding inputs, and -1 for the remaining features where zeros were meaningful values in the data. Padded time steps were then masked so that they were not considered by the network without it having to learn its irrelevance. Interaction sequences (in $\boldsymbol{X}_{\text{seq}}$) were truncated at 25 time steps, given that 95% of the sessions were smaller than this value. This also corresponds to the maximum impression length available (in $\boldsymbol{X}_{\text{imp}}$), and as embeddings are shared between item IDs in these two feature blocks, additional unnecessary padding is avoided. Because every individual metadata item attribute and filter was mapped to its own embedding vector, the attributes characterizing a single item in the impression lists and the active filters in a single session corresponded to variable length embedding sequences. To concatenate them with the remaining impression and session features, these embedding sequences were averaged, as done in [7], creating non-sequential unified sample representations. Each sequence was first padded with zeros and masked to obtain fixed sequences of length 112 (the maximum number of simultaneous item attributes and active filters). Metadata sequences, for instance, were then input to a Time Distributed Average Pooling layer (to average each of the available impression items' attributes) and then to a `Lambda` layer to re-mask any `NaN` (Not a Number) values resulting from averages over zero-only sequences.

VI. Model architecture optimization

The deep learning model's structural core was designed following a multitask transfer learning architecture [6], with

[7]The ratio of most to least frequent action counts dropped from 86.3 to 9.8.

[8]Destination searches, for instance, accounted for 40% of the first event's total interaction count, demonstrating a decreasing relative count trend with session progression also shared by actions including POI searches and changes in sort order.

[9]It was assumed that the ID spaces were immutable in the timespan considered. Vocabulary sizes - Item ID: 713 602, Attribute ID: 168, Action ID: 10, City ID: 20 268.

specialized branch modules processing the three main feature inputs separately ($\boldsymbol{X}_{\text{seq}}$, $\boldsymbol{X}_{\text{imp}}$, $\boldsymbol{X}_{\text{ses}}$, with parameter sharing limited to the embeddings), whose output representations are then combined, converging into a Multilayer Perceptron connected to the final dimensionality correcting softmax layer that produces the classifier's output. The highly conditional hyperparameter space fully described in Table IV defined a wide variety of possible model complexity and was optimized using a modified `Keras Tuner 1.0` [24] Bayesian Optimization with Gaussian Processes (BO GP), Algorithm 1, maximizing the MRR objective function.

Table IV. Hyperparameter space. (I) Inputs, (E) Embeddings, (B) Gated parameters, meaning they can become False. The ranges were defined from a combination of initial test runs, hardware limitation and typical literature values. Active M1 parameters are boldfaced. Children hyperparameters require active parents.

	Hyperparameters	Codes	Range	M1
Impressions	**Impression attributes (I)**	**H1**	{T, F}	**True**
	Item embedding (E)	**H2**	{2,...,50}	**2**
	Impression RNN type	**H3**	{GRU, S-GRU, Bi-GRU, Bi-S-GRU}	**Bi-S-GRU**
	Impression RNN units	**H4**	{10,...,250}	**250**
	Impression RNN dropout (B)	H5	[0,...,0.7]	**False**
	Impression Dense units (B)	H6	{10,...,250}	**False**
	Impression Dense dropout (B)	H7	[0,...,0.7]	**False**
	Metadata (I)	**H8**	{T, F}	**True**
	Metadata embedding (E)	**H9**	{2,...,30}	**3**
Interaction sequences	**Sequential codes (I)**	**H10**	{T, F}	**True**
	Sequential attributes (I)	**H11**	{T, F}	**True**
	Action embedding (E)	**H12**	{1,...,15}	**6**
	Sequential RNN type	**H13**	{GRU, S-GRU, Bi-GRU, Bi-S-GRU}	**Bi-S-GRU**
	Sequential RNN units	**H14**	{10,...,250}	**218**
	Sequential RNN dropout (B)	H15	[0,...,0.7]	**False**
	Sequential Self-attention type (B)	**H16**	{None, Add., Dot, Hierarch., Scaled}	**Hierarch.**
	Attention dimension	**H17**	{32,...,320}	**124**
	Sequential Dense units (B)	H18	{10,...,250}	**False**
	Sequential Dense dropout (B)	H19	[0,...,0.7]	**False**
Sessions	**Session features (I)**	**H20**	{T, F}	**True**
	Filters (I)	**H21**	{T, F}	**True**
	City embedding (E)	**H22**	{2,...,30}	**16**
	Session Dense units	**H23**	{10,...,250}	**142**
	Session Dense dropout (B)	**H24**	[0,...,0.7]	**0.33**
Joint MLP	**Out 3 Dense units (B)**	**H25**	{25;125,...,500}	**475**
	Out 3 Dense dropout (B)	**H26**	[0,...,0.7]	**0.10**
	Out 2 Dense units (B)	H27	{150,...,1000}	**False**
	Out 2 Dense dropout (B)	H28	[0,...,0.7]	**False**
	Out 1 Dense units (B)	H29	{2x Out 2 Dense units}	**False**
	Out 1 Dense dropout (B)	H30	[0,...,0.7]	**False**
Params	**Dense activation functions**	**H31**	{ReLU, PReLU, LeakyReLU}	**ReLU**
	Learning rate	**H32**	$[10^{-4},...,10^{-1}]$	**0.0012**

Algorithm 1: Bayesian Optimization with Gaussian Processes.

$\mathcal{D}_1 \leftarrow \varnothing$ ▷ *An initial collection of points can be used instead of an empty set*
for $n \in 1, ..., N$ **do**
  fit GP on observations to obtain $\mu_n(\boldsymbol{x})$, $\sigma_n(\boldsymbol{x})$
  select new hyperparameters $\boldsymbol{x}_{n+1}$ by optimizing acquisition function α, $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}} \alpha_n(\boldsymbol{x}; \mathcal{D}_n)$
  query objective function to obtain y_{n+1}
  augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\boldsymbol{x}_{n+1}, y_{n+1})\}$
  update statistical model
return model with best performing hyperparameter input

With a collection of input hyperparameter configuration points $\mathbf{x} = \boldsymbol{x}_{1:n}$, and noisy objective function observations $y = f(\boldsymbol{x}) + \epsilon$, the GP bayesian posterior's[10] mean and variance are given by:

$$\mu_n(\boldsymbol{x}) = \mathbf{K}(\boldsymbol{x}, \mathbf{x})[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}}\mathbf{I}]^{-1}\mathbf{y}, \tag{2}$$

[10]Assuming a Gaussian likelihood model given the observed data points and a zero-mean GP prior.

$$\sigma^2_n(\boldsymbol{x}) = \mathbf{K}(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{K}(\boldsymbol{x}, \mathbf{x})[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}}\mathbf{I}]^{-1} * \mathbf{K}(\mathbf{x}, \boldsymbol{x}). \tag{3}$$

Each entry in the covariance matrices $\mathbf{K}$ is given by the selected simplified *Matérn* kernel function with smoothness parameter ν set to 5/2,

$$k_{\nu=5/2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Big(1 + \frac{\sqrt{5}}{l} d(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{5}{3l^2} d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\Big) * \exp\Big(-\frac{\sqrt{5}}{l} d(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big). \tag{4}$$

The Upper Confidence Bound acquisition function was used:

$$\alpha_n(\boldsymbol{x}) = \mu_n(\boldsymbol{x}) + \beta_n^{1/2}\sigma_n(\boldsymbol{x}). \tag{5}$$

Default tuner σ^2_{noise} and β values of 0.0001 and 2.6 were used. Inactive hyperparameters in optimization runs were reset to default minimum values, as in [25]. The tuner optimizes the kernel's length scale l such that it maximizes the log marginal likelihood, with the L-BFGS-B algorithm.[11]

To cover the most amount of space in the limited time frame, the batch size was not tuned but instead fixed at 512 and early stopping patience[12] was set to 1. The Adam optimizer was used, with a tuned initial learning rate given by H32 (remaining hyperparameters such as moment estimate decay rates were left with default values).

The final process, which consisted of 1530 total runs, was divided into three modes:

1) **Random:** The bayesian optimization process can be initialized with an arbitrary number of data points. This implementation generated 180 random samples as initial training data for the tuner, corresponding to four times the dimensionality of the hyperparameter space (increased from the default three due to high conditionality).
2) **Bayesian:** The tuner iteratively applied Algorithm 1 for 1200 runs, from the random foundation.
3) **Top:** Due to the non-deterministic nature of neural networks, the top scoring hyperparameter set was not selected directly from the Bayesian section. Instead, the top 20 architectures were picked for five additional runs with different parameter initializations. The selection was then refined, with the top 10 being run an additional five times, for a more comprehensive performance overview.

In the final model architecture, described in Table IV and presented in Figure 6, the Hierarchical attention mechanism [26] in the sequential interaction branch outputs a weighted sum of the Stacked Bidirectional GRU's output $\boldsymbol{h}$:

$$\boldsymbol{c} = \sum_{j=1}^{n} \alpha_j \boldsymbol{h}^{(j)}, \tag{6}$$

$$\alpha_j = \text{softmax}(e_j), \tag{7}$$

$$e_j = \boldsymbol{u}_a^\top \tanh(\boldsymbol{W}_a \boldsymbol{h}^{(j)}), \tag{8}$$

where the weight vector $\boldsymbol{u}_a$ and matrix $\boldsymbol{W}_a$ are learned during the training process.

[11]If this algorithm does not converge within 20 restarts by default, the tuner draws a random sample from the hyperparameter space. The UCB acq. function was similarly optimized with L-BFGS-B, with a higher restart number of 50.

[12]Number of allowed epochs with increasing validation loss versus the minimum obtained in the run.

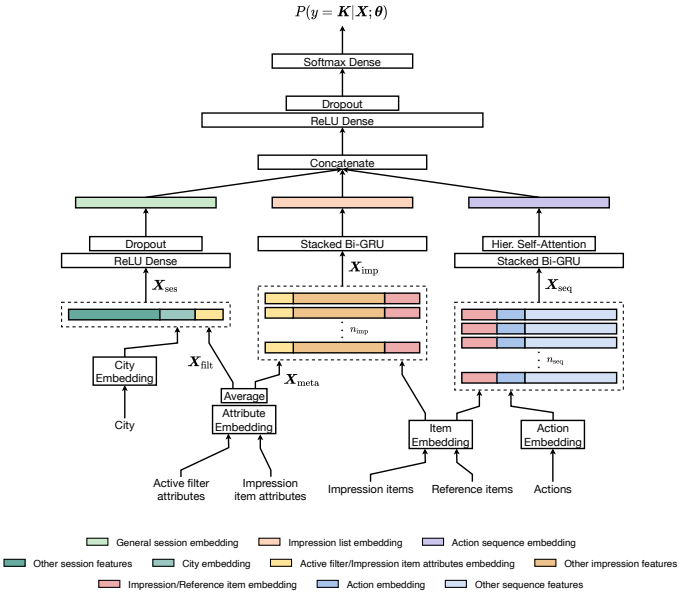

Fig. 6. Final model M1 architecture diagram. Components not to scale.

VII. Results

A. Architecture Optimization

The MRR results obtained in the optimization runs for the three different modes (Random, Bayesian, Top) are plotted in Figure 7 [13]

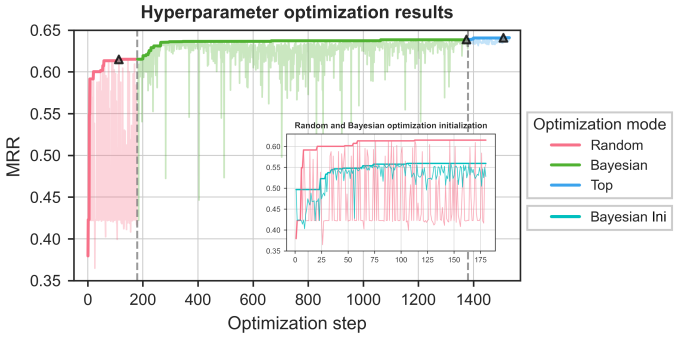

Fig. 7. Hyperparameter optimization and initialization results. Triangles mark the best runs in each mode.

While the Random mode got within 4.03% of the maximum value obtained, the MRR in its last 120 runs only increased by 0.29%. The Bayesian mode promoted a contrasting rapid performance increase of 3.29% in the following 85 runs, getting within 0.87% of the best result. Although the increase was much less significant during the mode's remaining runs, the fact that its best value was obtained close to the end signaled a possibility for further improvement, if time limits were not of concern. As additional advantages, this second mode combines significantly increased average performance values with an approximately four times smaller standard deviation, reflected in Figure 7's much lower MRR spread, where the observable lower peaks mostly correspond to the kernel optimization convergence failure runs mentioned in the setup, which draw random hyperparameter sets. The optimizer's exploratory ability is also maintained.[14]

To more directly assess the Random mode's impact, the first 180 runs were repeated using BO GP-UCB without space initialization. The Random mode's higher exploratory potential is favorable, providing a good foundation for future runs. With a constant UCB β, the more exploitative bayesian optimizer's performance is intimately tied to its initial most randomized runs, which, in this case, resulted in a 9% lower best initialization MRR. A possibly better performing fully bayesian alternative could make use of an adaptive β_n model, replacing the fully randomized section [27, 28].

B. Final Model Performance

As in the last optimization section, the M1 model was evaluated over ten runs with different initialization parameters but now taking advantage of the full RSC19 dataset. Lowering the batch size down to 128 was found to slightly improve the overall performance, with any further reduction only negatively impacting the training time. Early stopping patience was increased to five epochs in discarded initial runs monitoring the validation error on the previous section's optimization test set. It was noted that the model would consistently obtain the lowest error value around the second epoch, point at which the subsequent runs were stopped, producing the results presented in Table V.[15] The average performance values show increases of two (MRR) to three and a half percent (NLL) when compared to the optimization results.

Table V. Final M1 model architecture test results. Average and standard deviation values for 10 runs with different initialization parameters.

M1 Model	MRR	M. F1	Wt. F1	M. Rec.	M. Prec.	Wt. Prec.	Acc.	NLL
M1*	**0.65148**	0.47277	**0.52095**	**0.42011**	0.55582	0.53698	**0.53505**	**1.72953**
Average	0.65113	**0.47346**	0.51923	0.41682	**0.56471**	**0.53941**	0.53465	1.73079
O1 Delta	+2.01%	+3.44%	+2.69%	-	-	-	+2.54%	-3.54%
SD	0.00052	0.00116	0.00009	0.00413	0.01448	0.00595	0.00109	0.00118

With an arithmetic average MRR value of 3.81, the predictions additionally demonstrate a satisfactory concentration of 75% of the clicked items within the first four impression positions. M1*, the best weight combination, achieved an increased correct classification count but displays precision values below M1's averaged runs. The MRR values for the impression positions approximately reflect the class support's distribution but the F1-score displays an abrupt drop from 0.67 into a relatively stable average of approximately 0.46 right after the first position, due to the combined effect of the recall and precision distributions, naturally impacted by presentation bias. The recall displays a sharp decrease similar to F1's but bigger in magnitude, with values following the second class only achieving close to or less than half of the 0.83 maximum. In contrast, the best precision values are tied to lower positions, hinting that the models' predictions are more contextualized in this region.

Specific factors, including circumstances in which non-item based interactions correspond to last interaction sequence

[13]The average time per run was of 431 seconds, with CPU-based runs taking approximately 16 times longer than those completed on GPUs.

[14]Limiting a few impactful hyperparameters' values could significantly improve the Random mode's performance, a property that would require manual effort but that is exploited automatically in the Bayesian mode.

[15]Model training took an average of 309 seconds per epoch, while generating and evaluating the test set predictions, with a default batch size of 32, took an average of 0.53 milliseconds per sample.

events, contribute to skew the predictions towards upper positions, with clicks for the top position specifically increasing by 21.3% in these situations. Searching for new points of interest or changing sorting methods, for instance, can signal changes in the intent and objective of the session and introduce 'soft resets', meaning that the contextualization provided by the preceding sequence is negatively impacted. The extreme bias introduced by shorter interaction sequences in this dataset directly impacts the ranking performance. Although longer sequences are able to provide more predictive context, the expanded impression position possibilities and reduced number of training samples negatively affect the results.

C. Ablation Study

An ablation study was performed to assess the impact of individual model components and processing methods. Fourteen model variants based on M1 were tested under the full model setup, divided into three categories:

- **Model component removal** Single component removal, identified by the respective hyperparameter codes[16]. (**M2**) No impression features, H1; (**M3**) No interaction sequences, H10; (**M4**) No interaction sequence features, H11; (**M5**) No self-attention, H16; (**M6**) No session features, H20; (**M7**) No filter features, H21; (**M8**) No metadata features, H8; (**M9**) No joint dense layer, H25.
- **Model baselines** With the inability to directly compare full corpus session-based ranking recommenders to M1, two modified architectures which influenced its item embedding processing foundation, conditioned on the impression list items, were added as baseline references. (**M10**) Only interaction sequence and impression item embedding inputs, processed by the respective sequential blocks and joint MLP, inspired by GRU4Rec+ [10]; (**M11**) M10 with attention, inspired by NARM's local encoder [14].
- **Processing** Changes in data processing methods. (**M12**) No data augmentation, only last session clicks used for training; (**M13**) Class balancing method assigning more influence to minority classes in the cost function, with the class weights parameter as in [29]; (**M14**) Min-max normalization instead of Quant; (**M15**) Min-max normalization for ordinal features, Quant for the remaining.

Each configuration's results, corresponding to performance averages over five runs with different initialization parameters, are shown in Table VI. It can be noted that the overall most impactful model component corresponds, by a wide margin, to the impression features input. With macro and weighted F1 values 82.4% and 51.64% lower than M1, respectively, M2 is significantly worse classification-wise. Its also considerable ranking performance decrease of 27.25% is followed by M3's lack of sequential interaction input, with a much less significant (16.5 times smaller) drop.

On the other hand, M7 and M5's lack of filter features and self-attention mechanism, respectively, result in the smallest MRR decreases. Although the 0.7% ranking performance increase attributed to the Hierarchical attention implementation was not as significant as that of other model components, it promoted an increase in less supported class region predictions, leading to better recall values at the expense of slightly lower precision. Furthermore, the learned attention weight distributions provided insight into the model's sequential branch prediction contribution, improving recommendation interpretability. Two of the most interesting data characteristics captured by the mechanism include the relative higher importance of last sequential events[17] and the greater impact of specific item-based actions, including item searches and deal interactions. In addition, some sample examples showed skip-behavior awareness and the mechanism's ability to isolate interactions with specific items.

[16]Relevant hyperparameter H# codes are retrieved from Table IV

Table VI. Model ablation results averaged over 5 runs with different initialization parameters (10 for M1) and a batch size of 128. Best and second-best values are boldfaced and underlined, respectively.

Model	MRR	Delta (%)	M. F1	Wt. F1	M. Rec.	M. Prec.	Wt. Prec.	Acc.	NLL
M1	**0.65113**	**-**	**0.47346**	**0.51923**	0.41682	<u>0.56471</u>	<u>0.53941</u>	**0.53465**	**1.73079**
M2	0.47368	-27.25	0.08335	0.25107	0.08875	0.12400	0.23646	0.33909	2.42430
M3	0.64039	-1.65	0.46184	0.50972	0.41436	0.53666	0.52150	0.52199	1.77917
M4	0.64251	-1.32	0.46257	0.50870	0.40935	0.54868	0.52322	0.52223	1.77788
M5	0.64659	-0.70	0.46785	0.51421	0.40945	**0.56613**	**0.53992**	0.53046	1.74843
M6	0.64587	-0.81	0.46951	0.51518	0.41549	0.55599	0.53252	0.52811	1.77264
M7	<u>0.64719</u>	<u>-0.61</u>	0.46865	0.51565	0.41242	0.56258	0.53437	0.53025	<u>1.74149</u>
M8	0.64537	-0.89	0.46740	0.51239	0.41431	0.55236	0.53362	0.52878	1.75648
M9	0.64615	-0.77	0.46337	0.51081	0.40950	0.55229	0.53065	0.52554	1.75511
M10	0.45088	-30.75	0.06637	0.22062	0.07429	0.10170	0.20325	0.31191	2.50279
M11	0.45633	-29.92	0.06659	0.22060	0.07409	0.11903	0.21354	0.31525	2.48960
M12	0.63548	-2.40	0.45388	0.50331	0.40206	0.54027	0.52239	0.51871	1.79999
M13	0.59687	-8.33	0.43000	0.48901	**0.43655**	0.43245	0.51323	0.48034	2.01977
M14	0.63976	-1.75	0.44603	0.50290	0.38818	0.54916	0.52676	0.52114	1.78524
M15	0.64673	-0.68	<u>0.47272</u>	<u>0.51866</u>	<u>0.41932</u>	0.55976	0.53432	<u>0.53186</u>	1.76813

The simplest baseline M10 and M11 models seem to overrely on sequence-induced position information, obtaining only marginally better results than a model limited to top position predictions (POS, in the next Section **VII-D**). With a 30.75% ranking decrease, 7.1 and 2.35 times smaller macro and weighted F1s, compared to M1's average values, M10 was the worst performing configuration. The Hierarchical attention mechanism's inclusion in M11 improved its ranking performance by 1.21%.

Regarding processing methods, the increased training information provided by the data augmentation procedure was found to have a key positive effect in the overall performance of the model, corresponding to the second most important M1 ranking component. Although comparatively less influential in the ranking, proper data normalization was likewise found to have an important role in the results, showing clear benefits of the non-linear quantile transformation. Min-max usage in the whole dataset was linked to the third biggest MRR decrease when considering single M1 component edits. Its usage is disadvantageous even when the transformation is reduced to ordinal features. The attempt to balance the problem using the class weights parameter, which emphasizes minority predictions in the loss function, saw an increase in the ability of the classifier to find minority samples, as indicated by the 4.73% macro recall increase over M1's, but also a simultaneously significant decrease in every other metric, including the fourth biggest ranking-wise. Alternatives, such as different sampling methods, were not explored as most, especially binary extensions that balance according to the biggest or smallest class, might not be suitable for multiclass settings [30].

[17]This behavior is verified in other sparse datasets [16] and includes non-item-specific interactions, which generally prompt probability distributions loosely proportional to the class supports. Further data exploration helped justify this distribution in RSC19, by revealing that 37.9% of the clicked items correspond to the references of last item-based interactions.

D. Baseline Comparison

The development of suitable baselines to further contextualize M1's performance faced numerous challenges. User and utility-based solutions, including matrix factorization methods, are not suitable for sequential session-based environments and were not considered [19]. Neighborhood-based approaches are limited by their often large memory requirements, aggravated by RSC19's large item space dimensionality. Session-based kNN (s-kNN) and its variants have obtained decent results in some session-based problems [20] but their output scores based on item interaction occurrence and session similarity are oriented for next-item single-interactions predictions, predominant in e-commerce datasets. Since most impression items are not interacted with in RSC19's sequential events, s-kNN would not be able to efficiently produce recommendation scores for these without any type of modification. Therefore only simple, static baselines without learnable parameters were considered, based on those used in [8, 10, 14, 17] and the one provided by the challenge's organizers, (**CL-L**), which always recommends the most globally clicked items. The problem with CL-L is that it does not account for causality, using future click information to make predictions. This was fixed in (**CL**), which only uses each item's previously recorded clicks until the relevant click events' timestamps. (**S-CL**) applies the same concept to local session-based click information, using CL values to break ties. (**POP**) extends CL to make use of the remaining views, interactions and dwell time counts for a measure of global item 'popularity', with (**S-POP**) doing the same for S-CL. Given the much discussed bias towards top positions, (**POS**), which recommends only the top position's items, was added. Following the results of Section **VII-C**, the (**LAST**) model was created to recommend the last interaction sequence reference item, using S-POP values for non item-based actions. Finally, (**PRICE**) always recommends the cheapest impression item and a random predictor, (**RAND**), was added for reference.

Table VII. Baseline results and comparison to the best model M1*.

Baseline	MRR	M. F1	Wt. F1	M. Rec.	M. Prec.	Wt. Prec.	Acc.
M1*	**0.65148**	**0.47277**	**0.52095**	0.42011	**0.55582**	**0.53698**	**0.53505**
Delta	+13.72%	+11.88%	+6.82%	-3.21%	+34.71%	+8.71%	+10.64%
LAST	0.57288	0.42255	0.48770	**0.43406**	0.41260	0.49395	0.48358
S-POP	0.48889	0.24980	0.34504	0.24834	0.25176	0.34591	0.34446
POS	0.42420	0.01736	0.12030	0.04000	0.01109	0.07682	0.27717
CL-L	0.27397	0.09071	0.15718	0.09457	0.08867	0.17229	0.14914
S-CL	0.25488	0.09039	0.15016	0.09275	0.08914	0.16151	0.14376
POP	0.25439	0.07015	0.12223	0.07613	0.06900	0.15108	0.11035
CL	0.23116	0.07278	0.13066	0.07587	0.07138	0.14541	0.12283
PRICE	0.18965	0.05521	0.08882	0.07147	0.06135	0.13193	0.07691
RAND	0.15359	0.03246	0.04902	0.04040	0.04094	0.11612	0.04071

Obtaining only slightly better results than the random predictor, PRICE was the second-worst performing model. Although the provided baseline, CL-L, secured the fourth-best ranking result, it was underwhelming when compared to the 1.55 times bigger value required to enter the baseline top three. Unsurprisingly, the modified CL version performed worse, with relative drops of 15.6%, 19.8% and 16.9% for the MRR, macro F1 and weighted F1 respectively. S-CL's results were, however, only marginally worse than CL-L's, demonstrating the impact of local, session-based information in RSC19. POP's extension of CL resulted in a 10% MRR increase but a slight overall decrease in the remaining classification metrics, aside from weighted precision. Nevertheless, as with click information, the session-based S-POP version performed better. In fact, the local interaction popularity indicators proved to be important enough for the increase to be much more significant in both ranking (1.92 times bigger MRR) and classification (3.56 and 2.82 times multipliers for macro and weighted F1s), leading to the second-best baseline result. The benefits of having multiple evaluation metrics are apparent in POS' case, where the MRR (only 0.065 lower than S-POP, 1.55 times CL-L's) and also elevated accuracy might lead to misleading positive conclusions driven by the biased nature of the label distribution. With the exception of weighted F1, which is still decently influenced by the correct first-class predictions, the remaining classification values expectedly demonstrate the opposite, with results inferior to those of RAND. In the end, the best baseline result was achieved by LAST, with considerably better values than S-POP in every metric and even obtaining a narrowly better macro recall than M1*. Nonetheless, the remaining M1* delta values show that the deep learning model still significantly outperforms the baseline, whose MRR is lower than even the average returned by the Bayesian mode on the smaller optimization dataset. Regardless, given how simple the logic behind each of the baselines is, the obtained results are impressive and signal the possibility for potential competitive performances with further focus on more complex configurations.

E. RecSys Leaderboard

Applying [5]'s data split resulted in the ability to use their noted average difference between local and deployed submission results in the challenge's online test set of +1.7% to generate a final MRR estimate for M1* of 0.66255. This value is represented in the leaderboard's result distribution by position of Figure 8.

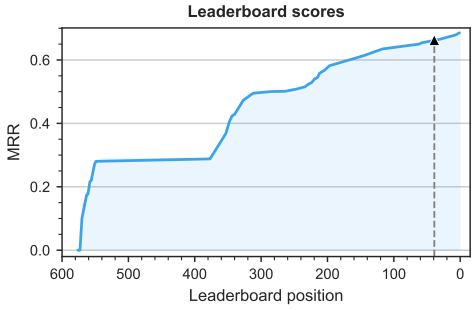

Fig. 8. RecSys19 leaderboard distribution and M1*'s estimated result.

The MRR distribution for the 575 submissions displays two significant value step concentrations starting at approximately positions 550 and 310. The former, more predominant one, corresponds to CL-L baseline results, which are in fact 4.83% higher than the local test set values obtained in the previous Section **VII-D**, while the latter more closely approximates the predicted S-POP result, with a 2.35% difference. M1*'s estimated performance would obtain the 39th position (one higher than M1's average), corresponding to the 93.5 and 85.7 percentiles when considering the full submission space and only values above CL-L's, respectively.

Although the gap to the top ten was still of a significant 2.11%, the obtained results are quite decent, especially when taking into account the unfeasible computational requirements,

model complexity, and feature generation focus of the best-performing entries, dominated by decision tree ensembles, versus M1*'s representation learning emphasis.[18]

Some of the feature-based strategies used by the preceding models, such as the generation of additional ranking, augmentation, aggregation and boolean inputs, which frequently rank the highest in decision tree model importance, could be used to most likely easily close the small 0.36% and 0.91% intervals to the top 30 and 20 positions without the need for architectural changes. Other transformations such as the input of squared, square root and log versions of continuous features, are reported to increase the network's expressive power in [7], could also be tested for the same effect. With respect to possible relevant structural changes for ranking performance improvement, besides ensemble implementations, the adoption of the transformer architecture, which returned the best neural network-based results in [32], would be a priority. The implementation of different attention types besides self-attention, such as those used in [5]'s 7th place solution, could also be used to enhance the model and enable it to use the sequential interactions to attend over the impression items directly, for instance. Factorization-based interaction layers like those in [34, 36] and residual connections, used in [16, 37], also seem to offer significant advantages in e-commerce datasets.

All of the enumerated changes would, however, still have difficulty placing the new model near the top ten, as the higher 2.11% MRR delta is also linked to the usage of non-causal information by the best performing models, namely the time difference between clicks and previous sequential events which is consistently reported as a top contributing feature [5, 31, 32, 34, 35], not used by M1.

VIII. Conclusions

This work encompassed the development of a deep learning re-ranking recommender system with self-attention for session-based environments, subject to an automated modular architectural bayesian optimization process, successfully accomplishing the initially proposed objectives.

While the predictive representations learned by the model are undeniably powerful, the re-ranking objective's applicability in this domain is limited by the lack of dynamic impression list presentation [35]. Unlike in YouTube or Amazon, where recommendations in dedicated video or product pages can change with every interaction, additional insight in trivago's recommendations can only be reflected after drastic interface events such as sort changes. Furthermore, although RSC19 most closely resembled a real-world scenario with the availability of metadata and other contextual features mostly disregarded in other e-commerce datasets (which often focus solely on item ID sequences), the lack of crucial information such as item-specific location data or interface details[19] hindered the modeling potential and prevented full ranking applications, which would have been arguably more useful.

[18]LogicAI's 1st place model [31] consisted of a 37 Multiple Additive Regression Tree (MART) ensemble with 250 different features, plus augmentations, trained on virtual machines with 96 vCPUs and 624GB RAM; Layer6's 2nd place result [32] was obtained by a linear blend of LSTM, Transformer, and Gradient Boosting Machine (GBM) models, with 330 features, running on a 256GB RAM machine with a Titan V GPU; the remaining entries in the top five [33–35] used a stacking GBM ensemble, an [36]-inspired neural network and GBM ensemble, and a GBM, Doc2Vec, MF and BPR hybrid, respectively, with the latter making use of 518 different features.

[19]Interface information was only available at click time and is challenging to estimate for other time steps. In some situations, interacted items were not present in impression lists even without any interface change signal, which constrained navigation evolution analysis, for example.

References

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[2] P. Knees, Y. Deldjoo, F. Bakhshandegan Moghaddam, J. Adamczak, G.-P. Leyson, and P. Monreal. RecSys Challenge 2019: Session-based Hotel Recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems*, RecSys '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6243-6/19/09. doi: 10.1145/3298689.3346974. URL https://doi.org/10.1145/3298689.3346974.

[3] J. Adamczak. RecSys Challenge 2019, March 2019. URL https://tech.trivago.com/2019/03/11/recsys-challenge-2019/. Last accessed on 2019-09-24.

[4] Our Product. URL https://company.trivago.com/our-product/. Last accessed on 2020-03-12.

[5] R. Gama and H. Fernandes. An attentive RNN model for session-based and context-aware recommendations: a solution to the RecSys challenge 2019. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618. URL http://www.deeplearningbook.org.

[7] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[8] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[9] B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, 2018.

[10] Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22, 2016.

[11] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

[12] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 241–248, 2016.

[13] E. Smirnova and F. Vasile. Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, pages 2–9, 2017.

[14] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[15] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang. 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*, pages 1831–1839, 2018.

[16] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[17] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[18] H.-H. Chen, C.-A. Chung, H.-C. Huang, and W. Tsui. Common pitfalls in training and evaluating recommender systems. *ACM SIGKDD Explorations Newsletter*, 19(1):37–45, 2017.

[19] M. Ludewig and D. Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4-5):331–390, 2018.

[20] M. Ludewig, N. Mauro, S. Latifi, and D. Jannach. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 462–466, 2019.

[21] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[22] A. F. Zuur, E. N. Ieno, and C. S. Elphick. A protocol for data exploration to avoid common statistical problems. *Methods in ecology and evolution*, 1(1):3–14, 2010.

[23] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[24] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al. Keras Tuner, 2019. URL https://github.com/keras-team/keras-tuner.

[25] J.-C. Lévesque, A. Durand, C. Gagné, and R. Sabourin. Bayesian optimization for conditional hyperparameter spaces. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 286–293. IEEE, 2017.

[26] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[27] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[28] F. Berkenkamp, A. P. Schoellig, and A. Krause. No-regret bayesian optimization with unknown hyperparameters. *arXiv preprint arXiv:1901.03357*, 2019.

[29] Classification on imbalanced data, 2020. URL https://www.tensorflow.org/tutorials/structured_data/imbalanced_data. TensorFlow Core API documentation. Last accessed on 2020-02-03.

[30] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[31] P. Jankiewicz, L. Kyrashchuk, P. Sienkowski, and M. Wójcik. Boosting algorithms for a session-based, context-aware recommender system in an online travel domain. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[32] M. Volkovs, A. Wong, Z. Cheng, F. Pérez, I. Stanevich, and Y. Lu. Robust contextual models for in-session personalization. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[33] Z. Wang, Y. Gao, H. Chen, and P. Yan. Session-based item recommendation with pairwise features. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[34] H. Kung-Hsiang, F. Yi-Fu, L. Yi-Ting, L. Tzong-Hann, C. Yao-Chun, L. Yi-Hui, and L. Shou-De. A-HA: A hybrid approach for hotel recommendation. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[35] M. Ludewig and D. Jannach. Learning to rank hotels for search and recommendation from session-based interaction logs and meta data. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[36] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1754–1763, 2018.

[37] S. Sun, Y. Tang, Z. Dai, and F. Zhou. Self-Attention Network for Session-Based Recommendation With Streaming Data Input. *IEEE Access*, 7:110499–110509, 2019.