Summary

Session-based recommendation considers the task of recommending items to users based on their interactions with the system during an ongoing session. The fundamental challenge of such recommender systems is thus to accommodate the user's current preferences in order to produce satisfactory recommendations.

In this thesis, we consider the session-based, next-item recommendation problem in the context of music where tracks must be recommended to a user based on what they have recently consumed, and what they are currently listening to. We attempt to push the current state-of-the-art further by considering two distinct directions for possible improvement: We first investigate whether awareness of implicit feedback during a session can help better capture short-term preferences and thereby improve recommendations. Specifically, we attempt to exploit the action of skipping a track in order to capture shifts in consumption motivations, or intent. Secondly, we consider how long-term preferences can be exploited in combination with short-term preferences—effectively transforming an otherwise session-based system into a session-aware system—in order to better understand a user and the natural development of their short- and long-term preferences.

In an effort to assess the merit of our proposed directions, we implement the state-of-the-art attentive neural network architecture, STABR. By combining bidirectional gated recurrent units with an attention mechanism, STABR generates recommendations for the currently active session based on the previous tracks and item features in the form of community-created tags associated with those tracks.

In order to address the first idea of employing implicit feedback in the form of skips, we customise STABR's gated recurrent units to be aware of whether a track has been skipped or not using additional embedded input. As to further evaluate the potential of skips as predictors for next-track recommendation, we extend a simpler, non-neural approach, namely session-based k-nearest neighbours, to also account for skips. Specifically, we use the similarity of sessions in terms of which tracks have been played to compute new recommendations. We then test the effect of only measuring similarity between sessions based on non-skipped tracks.

We address the second idea of incorporating long-term preferences by extending STABR with two additional gated recurrent unit networks, which generate hidden state representations of a user's previous sessions. We consider two variations for employing the long-term preferences of a user: The first variation uses the representation of previous sessions directly in the attention mechanism in order to assign attention weights based on the long-term preferences. The second variation uses the representation as an initial hidden state for the bidirectional gated recurrent units employed by STABR in order to generate new hidden states which are aware of a user's long-term preferences.

In order to determine the effectiveness of the extensions made to the STABR architecture, we evaluate using two distinct real-world datasets, each containing listening events from users of the online music database and social network, Last.fm. Additionally, we compare the results to a selection of baselines based on neural networks, matrix factorisation, and k-nearest neighbours. We find that, for the data used in our experiments, implicit feedback in the form of skips does not constitute a good predictor for session-based recommender. This is evident as both STABR and session-based k-nearest neighbours experience a drop in performance when extended to be aware of skips. More promising results are obtained by taking long-term preferences into consideration. While employing a long-term

preference representation in the attention mechanism of STABR does not improve performance, an improvement is observed when using the representation as an initial hidden state for the bidirectional gated recurrent units in STABR. While the improvement is small, we believe that it warrants further research in this direction.

In addition to the results related to skips and long-term preferences, we also find that complex neural approaches such as STABR may not always be justified over simpler approaches such as k-nearest neighbours. Indeed, while architectures such as STABR consistently perform well in terms of mean reciprocal rank, they are occasionally outperformed in terms of hit rate. As such, what constitutes the best approach both depends on the data at hand, but also in large part on which practical problem the system ought to address.

Exploring Skips and Long-Term Preferences in Session-based Music Recommendation

Peter Vergerakis
Department of Computer Science
Aalborg University
Aalborg, Denmark
pverge15@student.aau.dk

Joachim Valdemar Yde
Department of Computer Science
Aalborg University
Aalborg, Denmark
jo@ch.im

ABSTRACT

While traditional recommender systems aim to help users find items that they may generally like, session-based recommenders face a unique challenge in that they are required to recommend items a user may enjoy in the immediate or near future. Thus, considerations of context and short-term preferences become crucial in the pursuit of computing satisfactory recommendations. In recent research, neural-based approaches leveraging recurrent neural networks and attention mechanisms have been successfully applied in fields such as e-commerce and music. Indeed, with the increasing popularity of on-demand music streaming services, session-based recommendation has become more pertinent than ever. In this paper, we implement the state-of-the-art neural attentive-based architecture STABR, and propose two extensions: One focused on short-term preferences through implicit feedback in the form of skips, and another focused on long-term preferences using prior sessions. We conduct experimentation on two distinct datasets, and compare STABR and our extensions to a range of alternative techniques in order to discuss the generalisability of STABR, the efficacy of our extensions, and to illuminate the performance of complex neural models compared to simpler approaches. We find that implicit feedback in the form of skips adds more noise than signal. We further find that accounting for both short- and long-term preferences in the same model using prior user sessions produces promising results warranting further research. Additionally, our results suggest that complex neural models do not consistently outperform simpler approaches such as k-nearest neighbours in session-based music recommendation.

1 INTRODUCTION

The continued popularity of on-demand music streaming services exposes an ever-increasing number of users to fast-expanding libraries of readily-available music. Applying recommender systems has proven an effective solution to help users comfortably explore such complex information landscapes [3, 50]. However, compared to other fields commonly served by recommender systems such as e-commerce and movies, music is unique in both its abundance and disposable nature as well as its consumption. Indeed, the vast majority of users consume dozens of tracks in succession as they go about their day [8]. This consumption, moreover, commences in different contexts, both spatial and temporal, e.g. during an early-morning jog, a late-night reading session, or whilst commuting [44]. It is thus only reasonable to assume that sessions differing in context also differ in content; effectively making the individual tracks of a specific session and their order important indicators of the current context and intent [12]. Yet, conventional recommender systems typically disregard such sequential information as they seek to infer static relevancy between users and items. Systems based on matrix factorisation, for instance, may be effective at modelling the general preferences of a given user by analysing their past interactions with the system, but they do not model the order of these interactions [15, 22]. This static representation is problematic because it fails to accommodate both short- and long-term preferences, and, as a direct consequence, does not adequately account for how the preferences of a given user naturally evolve over time [23, 47].

In contrast to conventional recommender systems, session-based recommender systems seek to predict the next item of interest by modelling a user's short-term preferences using their current session. This is typically achieved by exploiting implicit feedback given to the system by the user, e.g. in the form of clicks [29]. The modelling choices for these recommenders include both Markov chains, e.g. the Factorizing Personalized Markov Chain (FPMC) model [32], and various recurrent neural networks (RNNs) models [5, 27, 31, 45]. However, where solutions based on Markov chains only model local sequential patterns between adjacent interactions [5], RNNs can successfully model multi-step sequential behaviours [15, 31, 37]. In either case, such session-based recommenders typically embed a user's long-term preferences into a static representation [5]. As such, this approach only addresses half the shortcomings facing conventional recommender systems because long-term preferences present a varying degree of importance depending on the current short-term, sequential dynamics [43, 48]. In the context of music, the relative importance of long-term preferences depends on the current intent of the user. A user might have a session in which they wish to discover new artists as opposed to listening to known ones; or they might have a session in which they wish to merely enjoy old-time favourites. Whilst there is no financial loss or significant loss of time as a result of a poor track being recommended—compared to a poor book or movie recommendation, for instance—the flow in which the user may find themselves can be disturbed, causing annoyance or frustration with the service. A good session is thus, in many ways, a session which runs passively and succeeds in satisfying the evolving desires of the user in the given context.

Indeed, in contrast to other sequential or session recommendation scenarios such as e-commerce, the shift from one item to the next does not necessarily constitute an active action by the user as most streaming services allow the transition between tracks to pass passively. In fact, the only definitively active actions a user generally makes during a given session—beyond starting and ending the session—are those which disrupt the playback. This includes pausing, replaying, and skipping; actions which can be considered neutral, positive, and negative signals, respectively. While the sentiment of these signals can be more or less precisely determined, their underlying intents cannot. Skipping a track should not be unequivocally interpreted as the user not liking that specific track, for instance: The current context might prompt the user to skip a track they otherwise would have enjoyed in a different setting.

Given this nature of music sessions, it is reasonable to assume that any system intended to produce accurate next-item recommendations should accommodate such non-linear and non-monotonic relationships as to better capture the consumption motivations of individual users [20, 46]. In this regard, RNNs feature great flexibility in capturing such relationships. However, traditional RNNs face issues handling long-range dependencies due to the problem of vanishing gradients, ultimately reducing their efficacy in dealing with long-term preferences [2, 16]. This limitation has been partially overcome using both long short-term memory units (LSTMs) [17] and gated recurrent units (GRUs) [6], achieving great success in fields such as machine translation and image processing [7, 37]. Ultimately, however, the performance of both LSTMs and GRUs will eventually deteriorate as the length of the input increases [1, 6]. In order to address this shortcoming, Bahdanau et al. proposed attention-based encoder-decoder networks [1]. They showed that the introduction of an attention mechanism—carefully selecting hidden states across all time steps depending on what is deemed the most relevant for the current step—enables models to better accommodate long-range dependencies.

In this paper, we explore possible new research directions within attention-based, next-item music recommendation. We implement the current state-of-the-art attention-based recommender system, STABR, as proposed by Sachdeva et al. [33], and examine possible extensions seeking to (a) incorporate active user actions; and (b) better accommodate long-term preferences. Specifically, we consider the following extensions:

- A customised, contextual GRU network capable of incorporating implicit feedback from users in the form of skips as to better capture consumption motivations.
- Two distinct variations of modelling a user's long-term preferences by explicitly considering previous sessions either directly or indirectly in the attention mechanism.

In order to validate the merit of these extensions, we conduct experiments comparing them and STABR to a range of baselines relying on different techniques of varying complexities on two distinct datasets. Based on these experiments, we discuss the efficacy of the extensions as well as which circumstances might warrant complex model architectures, and which might not. Additionally, in the context of session-based music recommendation, we are, to the best of our knowledge, the first to:

(1) Experiment with implicit feedback and user actions such as skips in an offline learning setting; and
(2) experiment with long-term preferences for item features such as genre descriptors.

The rest of the paper is structured as follows: Section 2 presents selected related work; section 3 outlines our working hypothesis, a formal problem definition, introduces STABR, and presents our proposed extensions; section 4 introduces our two datasets and outlines our pre-processing steps; section 5 presents our baselines, relevant parameters for training and testing, as well as presents and discusses our results; and section 6 concludes our work and outlines relevant future work.

2 RELATED WORK

We summarise related work in two selected areas; sequential recommender systems and attention-based models. In both areas, we discuss general advances as well as advances specifically within the field of music recommendation.

2.1 Sequential Recommendation

According to Quadrana et al. [31], session-*based* recommendation as described in the introduction denotes the problem in which only the current sequence of actions by a given user within some specified interval is known. Since listening to music is a naturally reoccurring event for the vast majority of users, we are typically dealing with session-*aware* recommendation in which both the current and prior sessions are known. That is, session-aware recommender systems generally seek to predict what happens next by modelling a user's short-term preferences using past sessions alongside implicit feedback given to the system by the user during the current session; essentially exploiting any dynamic, sequential patterns present in the user's behaviour. However, because any system supposed to be session-aware must also account for the instance in which only a single session is available, we will focus primarily on the problem of session-based recommendation.

Appreciating the importance of different spatial and temporal contexts when working with music recommendation, Gupta et al. [12] seek to explicitly model the short-term preferences of a user by incorporating track-specific, community-created tags (hereinafter tags) obtained from Last.fm[1]. They go on to show that while users often have implicit preferences when listening to music, those preferences are not necessarily static across different contexts. As such, Gupta et al. assume that a context can change both during a session and between individual sessions. In order to accommodate this, they propose the notion of sub-sessions which represent slices of sessions wherein the preference of the user does not change significantly. Sub-sessions are extracted by analysing the associated tags of each individual track and subsequently comparing them to those of neighbouring tracks. Having extracted a single sub-session, Gupta et al. proceed to mine their complete training data in order to identify the most similar sub-sessions from which they compute candidate tracks and eventually final predictions.

A fundamentally different approach also seeking to model the dynamic preferences is proposed by Rendle et al. [32], who attempt to predict next-purchases in an e-commerce setting. Using personalised Markov chains over sequential user data, their FPMC model allows each transition (i.e. user action) to be influenced by actions of similar users, similar items, and similar actions. In fact, Markov models (including both Markov chain (MC) and Markov decision process (MDP)) constitute a major category in the field of session-aware recommenders according to both Quadrana et al. and Ludewig and Jannach [27]. Rendle et al. thus propose merely one of several Markov model-based approaches: In essence, a Markov model considers sequential data as a stochastic process over discrete random variables and limits the dependencies of the process to a

[1]https://www.last.fm/api/ - an online music database and social network.

finite history. The essential challenge when using Markov models, therefore, is determining how many prior interactions should be considered when predicting the next. In the case of Rendle et al., this is determined by their factorisation component and allows them to have a higher quality transition graph. An earlier solution also in the field of e-commerce using a MDP model was proposed by Shani et al. [35], who devised a number of heuristics alongside a custom predictive model to ensure a reasonable initialisation. Other work such as Aghdam et al. [18] relies on hierarchical hidden Markov models, and other work again by Wang et al. extends the original FPMC model by including a hierarchical structure to learn user representations [45]. In any case though, the fundamental shortcoming of Markov models in session-based recommendation is their inability to adequately deal with long-range dependencies and limited ability to model the inherently complex dynamics often present in user-action sequences [5, 27, 31].

This conclusion makes RNNs a compelling solution to session-based recommendation purely based on their modelling power. In short, RNNs are distributed real-valued hidden state models with non-linear dynamics—something which effectively enables them to model sequentially ordered data whilst accounting for both long- and short-term dependencies [27, 31]. The inherent advantage of RNNs, as argued by Hidasi et al. [14], is their ability to model sessions in their entirety and thus provide better recommendations compared to traditional methods. Indeed, RNNs are able to encode both the additional contextual information proposed by Gupta et al., the short-term dependencies captured by Markov models, as well as accommodate long-term dependencies which neither Gupta et al. nor Markov-based models account for.

However, as noted in the introduction, traditional RNNs are not perfect and ultimately face issues in adequately dealing with long-range dependencies due to the vanishing gradients problem [2]. Consequently, in the context of session-based recommendation and other sequence modelling problems, extensions such as LSTMs or GRUs become necessary given enough data. Indeed, as noted by Vaswani et al. [42], LSTMs and GRUs have firmly been established as state-of-the-art approaches in sequence modelling in fields such as machine translation, speech recognition, and image processing. Accordingly, Hidasi et al. [15] pioneered the idea of employing RNNs for session-based recommendation to deal with data sparsity problems with their GRU4REC architecture. Specifically, Hidasi et al. sought to capture how a given session evolves over time. As the name implies, Hidasi et al. rely on GRUs in their network; depending on the input, one or more GRUs are strung together followed by one or more feed-forward networks between the last GRU and the final output. Due to the sound performance of this architecture, it has subsequently been extended by others, including Tan et al. [38], who proposed both data augmentation to improve performance and introduced a method to account for shifts in the input data distribution, and Jannach et al. [21], who incorporated k-nearest neighbour (kNN). Similarly, Hidasi et al. recently extended their original GRU4REC architecture by employing an improved loss function [14]. While the continued work on GRU4REC means that it remains a sound solution to several session-based and session-aware recommendation problems, it is important to note that in pursuit of capturing how a session evolves over time, GRU4REC, and its subsequent extensions, employ session-parallel mini-batches which means that the architecture is unable to handle unseen data.

In our case, the modelling power offered by RNNs in conjunction with extensions such as GRUs makes a compelling solution in wanting to accommodate both long-term dependencies and user actions. However, GRU4REC does not actively account for information such as tags the way Gupta et al. do, for instance, nor is it necessarily capable of handling very long-range dependencies.

2.2 Attention-based Recommendation

In recent years, attention-based models have become increasingly popular solutions to various session- and sequential-based recommendation problems—both as standalone applications using self-attention, and in conjunction with existing recurrent networks [1, 33, 42]. The central problem solved by an attention mechanism is modelling dependencies without regard to their distance in the network's input and output; that is, attention enables a network to refer to its input sequence directly as opposed to encoding input into a fixed-length vector. This not only addresses the shortcomings of LSTMs and GRUs discussed in sections 1 and 2.1, it further proves powerful in capturing sequential behaviour: Li et al. [25], for instance, propose a neural attentive recommendation machine (NARM) that combines a GRU network and an attention mechanism in order to model a user's sequential behaviour. Specifically, NARM uses the most recent hidden state from its GRU network in combination with hidden states from prior time steps in the same session to learn which items are important for the next-item recommendation. According to Li et al., the attention mechanism proves vital as it helps better capture the intent of the user—something which enables NARM to outperform general RNN models.

However, where NARM succeeds in modelling the short-term preferences of users, it does not consider the long-term preferences that might exist across multiple sessions. To this end, Chen et al. [5] propose a dynamic co-attention network for session-based recommendation (DCN-SR) in an e-commerce setting. Similar to NARM, DCN-SR employs a GRU network to capture the short-term preferences of users based on their current session. Additionally, a multilayer perceptron is utilised in parallel to the GRU network in order to capture long-term preferences from the user's historical interactions. Using a co-attention network, correlations between the long- and short-term preferences of the user are then explored. By considering the interplay between current and historical interactions, DCN-SR is able to provide better context for assigning relative importance to each item in the current session. DCN-SR further extends the traditional GRU network with contextual information in order to explicitly consider the actions—clicks, add-to-cart, and purchases—that the user performs for each item in a session.

Tang et al. [39] adopt a different approach in order to model both the long- and short-term interests of users for sequential recommendation. Specifically, they propose a neural multi-temporal-range mixture model (M3), which employs three distinct encoders in parallel to capture the preferences of users at different temporal ranges. Item co-occurrences are used for the first encoder in order to learn patterns of items occurring immediately after one another. An RNN network is then used for the second encoder in order to capture patterns occurring in a longer, but still relatively short, temporal range. Finally, a self-attention model is used as the last encoder—something which allows M3 to capture patterns in unlimited temporal ranges. The last item in the user's history is used as the query in order for the model to calculate the relative importance of previous historical items. Through a gating mechanism, M3 is able to use different combinations of the three encoders in order to address different recommendation scenarios.

In the realm of music, attention-based models have also been exploited for the session-based recommendation task. Huang et al. [19], for instance, propose an attention-based short-term and long-term model (ASLM) for next-item recommendation. Similar to [5], short- and long-term preferences are the key aspects considered in the architecture. An encoder-decoder architecture is used to model short-term preference to allow ASLM to produce recommendations over multiple time steps. Specifically, a bidirectional LSTM network is used as the encoder, and a GRU network is used for decoding. Similar to [25], an attention layer sits between the two, allowing the encoder-decoder to better capture the variation in short-term intent. The attention mechanism uses the last hidden state from the decoder alongside the hidden states from the encoder to determine which items to pay more attention to when computing the next recommendation. In contrast to the work of Li et al., however, ASLM also considers the influence of a user's long-term preference. A series of vector representations of the user's most recent sessions is fed to a GRU network. The final hidden state of this network is then used as the initial hidden state of the short-term preference encoder-decoder, thus allowing ASLM to account for both preferences when computing next-item recommendations.

Sachdeva et al. [33] take a different approach to session-based music recommendation by only considering the short-term preferences which exist in a session. Specifically, their proposed song and tag attention based recommender system (STABR) relies on using features of tracks to extract information about the preferences of a user. Following the idea from Gupta et al. [12] outlined in the previous section, tags are used for this purpose. STABR employs two nearly identical components consisting of a bidirectional GRU (BiGRU) network and an attention mechanism. While one component considers the actual tracks of a session, the other component looks at the tags associated with each track. Sachdeva et al. show that by explicitly considering features of tracks in the form of tags, STABR is able to achieve better results than its feature-ignorant counterpart, SABR, which in turn outperform all chosen baselines. Indeed, to the best of our knowledge, STABR represents the current state-of-the-art solution in session-based music recommendation. STABR thus also presents an interesting take on the problem that is session-aware recommendation considering its ability to produce good recommendations relying merely on short-term preferences and the contextual information available through tags. Notably, however, neither STABR nor SABR considers previous sessions or the user's actions in the given sessions.

3 APPROACH

Our working hypothesis is that whilst sequential information is paramount in accommodating short-term preferences during a singular session, accounting for implicit feedback alongside the relative importance of a user's prior interactions—i.e. long-term preferences—yields a more accurate user representation. This hypothesis is really twofold, ultimately appreciating the inherent intricacies of short- and long-term user preferences.

We speculate that by explicitly considering implicit feedback such as skips, the current intent of a user may be learned such that the development of that user's short-term preferences can be better gauged. We choose to focus on skips because pausing is more likely to happen due to exogenous variables than the music itself, and replaying a track cannot be considered a shift in context nor intent. Similarly, by explicitly considering prior sessions, the development of the current consumption motivations—i.e. intent—and thus short-term preferences might be better captured.

Consider the following scenarios: A user is listening to music while they are cooking dinner with a friend. During this time, they are enjoying some up-beat dance music. However, as the meal is finished and they sit down to eat, the user changes—i.e. skips—to something more ambient to allow room for conversation. Accordingly, with the shift in context, the user's short-term preferences in genre follow. In another scenario, the same user is listening to alternative rock, a genre that the user is familiar with and generally enjoys. After a while, the user is feeling adventurous and decides to explore some new alternative rock artists a friend mentioned in passing. The user thus queues a selection of tracks they have not previously listened to. As the user assesses the selected tracks, they skip any track that they dislike, effectively resulting in more skips than a typical listening session. In this scenario, the short-term preferences have not necessarily changed in terms of genre, but rather in terms of scope and artist familiarity within the given genre. While the shift in preference in the first example can be observed based on the abrupt change in genre, the same is not the case for the second example as the genre remains unchanged, but the increased number of skips suggests a changed intent.

In fact, the second scenario might have more lasting effects as the new tracks played in full during an explorative session could impact both current short-term preferences as well as the long-term preferences of the user. What music best suits a given context is naturally wholly subjective, and one user might prefer one type of music during weekends and something else during the week. Importantly, however, this preference, as shown by Gupta et al. [12], is likely to evolve over time. Similarly, the preferences within two different genres of a given user can evolve independently. It is thus important to accentuate how we are not trying to predict *why* a shift in context nor intent happens, but rather that it happens and which direction the change is in. This is something we speculate can be achieved by also accounting for genre identification in the form of tags across both short- and long-term preferences.

In light of our related work section, we believe that extending the STABR architecture proposed by Sachdeva et al. [33] is a natural foundation given both STABR's modelling power and its position as a state-of-the-art attention-based music recommender. It is, to the best of our knowledge, the only state-of-the-art architecture which considers item features such as tags—something which really encapsulates both genres, moods, and other relevant topics that can help categorise tracks [24]. Additionally, despite STABR being exclusively session-based, and thus only focused on short-term preferences, the architecture consists of a BiGRU-component and an attention mechanism which both can accommodate long-term dependencies—something which proved beneficial in other implementations, e.g. ASLM by Huang et al. [19]. This architecture thus allows us to directly evaluate the importance of both implicit feedback in the form of skips, as well as the relevance of long-term dependencies for session-based music recommendation. Thus, our work constitutes an exploratory analysis of possible extensions towards pushing the state-of-the-art in the session-based music recommendation further.

3.1 Problem Formulation

Let $U = \{u_1, u_2, ..., u_{|U|}\}$ be the set of users, $V = \{v_1, v_2, ..., v_{|V|}\}$ the set of tracks, and $G = \{g_1, g_2, ..., g_{|G|}\}$ the set of tags, where $|U|$, $|V|$, and $|G|$ denote the total number of unique users, tracks, and tags, respectively. A given track is associated with a set of tags such that $G_i = \{g_{i,1}, g_{i,2}, ..., g_{i,j}\}$ is the set of tags associated with track v_i, and $g_{i,l} \in G$. The listening event history H of a user u is a list of tracks with corresponding timestamps t: $H^u = \{(v^u_1, t^u_1), (v^u_2, t^u_2), ..., (v^u_{|H^u|}, t^u_{|H^u|})\}$, where $v^u_i \in V$ and t^u_i is the timestamp for listening event i of user u, such that $t^u_{i-1} < t^u_i < t^u_{i+1}$. The listening event history of a user can be aggregated into sessions, $S^u = \{s^u_1, s^u_2, ..., s^u_{|S^u|}\}$, based on the timestamps of listening events such that the time between the last timestamp of session s^u_i and the first timestamp of session s^u_{i+1} is at least x, where x denotes a given threshold. The recommendation task then becomes: Given the listening event history, H^u, of user u, the tracks in their current session, $s^u_{|S^u|+1} = \{v^u_{|H^u|+1}, v^u_{|H^u|+2}, ..., v^u_{|H^u|+i-1}\}$, and the tags associated with each track, predict the next track, $v^u_{|H^u|+i}$.

Or, in more informal terms: Assuming a set of users, tracks, and tags, we have a collection of sessions for each user consisting of a subset of tracks and their associated tags within a specified interval. Given the current session of an arbitrary user, predict the next track in this session.

3.2 Preliminaries

Before presenting our extensions to STABR, we introduce the original architecture. While a detailed description is provided in the original paper [33], we include one as well for completeness and to detail the assumptions we have had to make in pursuit of increased reproducability[2]. Indeed, Dacrema et al. [10] explicitly list STABR as a model they were unable to reproduce with reasonable effort.

An overview of the STABR architecture can be seen in figure 1. The core architecture consists of two components: For each track occurring before the track to be predicted in the active session of some user u, the first component receives the one-hot encodings of the tracks, and the second component receives the one-hot encodings of the tags associated with each of the tracks. An embedding of the one-hot encodings constitutes the first layer of the architecture in which the encodings are mapped to a vector space, such that

$$v'_i = E^v \cdot v_i \tag{1}$$

where v'_i is the embedded representation of track v_i, $E^v \in \mathbb{R}^{d \cdot |V|}$ is the embedding layer for tracks, and d is the size of the embedded track vector. Similarly, embeddings of tags are obtained as follows:

$$g'_{i,j} = E^g \cdot g_{i,j} \tag{2}$$

[2]All code is available at: https://github.com/x775/SW10

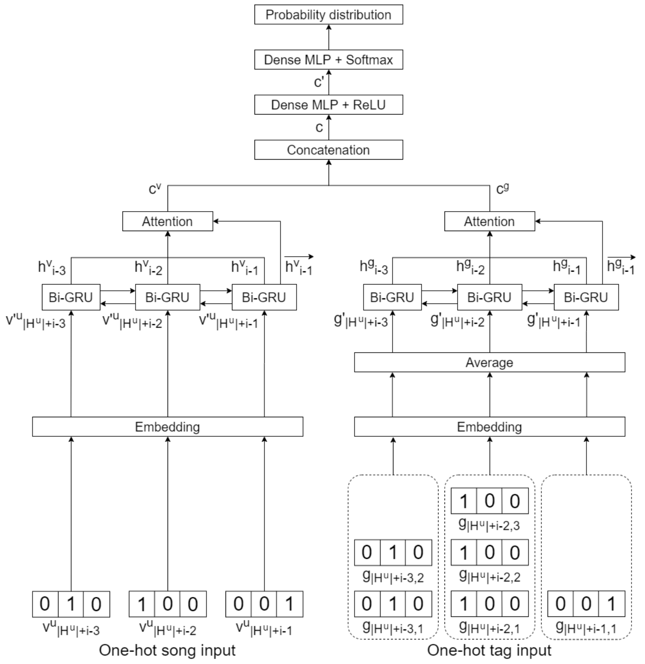


Figure 1: Overview of the STABR architecture.

where $g'_{i,j}$ is the embedded representation of the j-th tag of track i, $E^g \in \mathbb{R}^{d' \cdot |G|}$ is the embedding layer for tags, and d' is the size of the embedded tag vector. Since a track can have multiple tags, the embedded tag vectors for track i are averaged to obtain a final tag vector:

$$g'_i = \frac{1}{n_i} \sum_{j=1}^{n_i} g'_{i,j} \tag{3}$$

where g'_i is the final embedded vector for tags of track i, and n_i is the total number of tags associated with track i. The resulting track and tag embedding vectors are fed to two BiGRU networks.

As such, for each track and tag vector in the active session, two hidden states are obtained. These hidden states are then fed to the attention layers in order to produce context vectors which are weighted sums of the hidden states. Specifically, the context vector for track embeddings c^v in the active session is computed as:

$$c^v = \sum_{j=|H^u|+1}^{|H^u|+i-1} \alpha_j h^v_j \tag{4}$$

where h^v_j is the hidden state representation for track j obtained by concatenation of the forward and backward hidden states, $\overrightarrow{h^v_j}$ and $\overleftarrow{h^v_j}$. $|H^u|$ is the listening event history of user u. In a similar fashion, the context vector for tag embeddings c^g is obtained as follows:

$$c^g = \sum_{j=|H^u|+1}^{|H^u|+i-1} \beta_j h^g_j \tag{5}$$

Notably, no definitions of weight vectors α and β are supplied in the original paper by Sachdeva et al. [33]. As such, we assume a definition similar to that of the NARM architecture presented by Li et al. [25], which in turn is based on additive style attention detailed in the original attention paper by Bahdanau et al. [1]. Thus, we define α and β as follows:

$$\alpha = \mathrm{softmax}(\alpha') \tag{6}$$

$$\alpha' = w^v \tanh(Z^v P^v + b^{zv} + Q^v \overrightarrow{h^v} + b^{qv}) + b^{wv} \tag{7}$$

$$\beta = \mathrm{softmax}(\beta') \tag{8}$$

$$\beta' = w^g \tanh(Z^g P^g + b^{zg} + Q^g \overrightarrow{h^g} + b^{qg}) + b^{wg} \tag{9}$$

where w is a weight parameter vector, Z^v, Z^g, Q^v, and Q^g are weight parameter matrices. b^{zv}, b^{qv}, b^{zg}, b^{qg}, b^{wv}, and b^{wg} are bias parameter vectors. $\overrightarrow{h^v}$ and $\overrightarrow{h^g}$ are the forward pass hidden states of the track and tag embedding vectors at the last time step in the current session, respectively. P^v and P^g are matrices of all hidden state representations for tracks and tags, respectively, such that $P^v \in \mathbb{R}^{n \cdot d}$, and $P^g \in \mathbb{R}^{n \cdot d'}$, where n is the number of hidden states, and d and d' is the size of each hidden state for tracks and tags, respectively. Softmax is applied so that weights in α and β sum to 1. With this definition, α and β can be considered to model the alignment of the hidden state representations of each time step in the active session and the hidden state of the track played immediately before the track to be predicted. Thus, in effect, the attention layer helps the model focus on specific tracks and tags in the current session which are deemed more important for predicting the next track. The two context vectors, c^v and c^g, are then concatenated to obtain a final context vector, c, which is fed to a dense layer:

$$c' = \mathrm{ReLU}(W_1 c + b_1) \tag{10}$$

where W_1 is a weight parameter matrix, and b_1 is a bias parameter vector. The final output is obtained from a second dense layer followed by a softmax function to obtain probabilities of occurrence for each track in the set of tracks, V:

$$O = \mathrm{softmax}(W_2 c' + b_2) \tag{11}$$

3.3 Proposed Extensions

Based on our hypothesis outlined in section 3, we propose two extensions to the STABR architecture. We explain our intuition behind the extensions and their designs in sections 3.3.1 and 3.3.2.

3.3.1 Skip Extension. Skips exist as implicit feedback during a session which we hypothesise can be exploited to better capture the intent and the short-term preferences of a user during a session. A naive approach to employing skips would be to only consider tracks in an active session which have not been skipped. However, this approach would also discard potentially valuable information that can help explain what genres or tracks a user is not interested in during a specific context. Furthermore, as outlined in the two scenarios given in section 3, it may be that a user skips a track because their short-term preference has shifted from one genre to another. However, it could also be that their preferences remain the same, but their intent has shifted from passive listening to active exploration. In such a context, a skip can signify that the user is satisfied with the current genre, but not the specific track.

With these considerations, we draw inspiration from Chen et al. [5], and extend both the tracks and tags component of STABR by employing contextual GRUs instead of traditional GRUs. This enables the architecture to capture skips during a session and learn to interpret their meaning and significance. The structure of the contextual GRU network can be seen in figure 2. In addition to the track or tag embedding and the previous hidden state, the contextual GRU cell is modified to also take as input an embedding of a user action a:

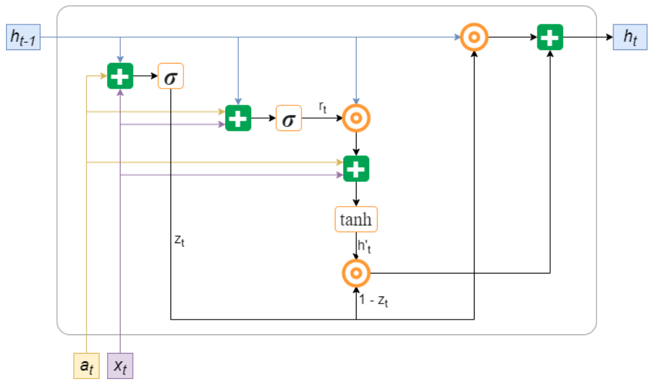


Figure 2: Structure of the CGRU network. t represents a specific time step, and x_t is either an embedded track or tag vector. The arrows at the end of all lines show the flow of information through the cell.

$$a'_i = E^a \cdot a_i \tag{12}$$

where a'_i is the embedding of the action performed for the i-th track in the current session, $E^a \in \mathbb{R}^{d^a \cdot |A|}$ is an embedding layer for actions, d^a is the length of the embedded action vector, and $|A|$ is the total number of possible actions. We consider two possible actions: "*skipped*" and "*not skipped*". The new hidden state is obtained from the previous hidden state and the candidate hidden state

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{13}$$

where h_t is the new hidden state at time step t, h_{t-1} is the previous hidden state, $\odot$ is the Hadamard product operation, and z_t is the update gate given by

$$z_t = \sigma(W^z x'_t + b^{xz} + U^z h_{t-1} + b^{hz} + Q^z a'_t + b^{az}) \tag{14}$$

where x'_t is the embedded track or tag input, and a'_t is the embedded skip action. W^z, U^z, and Q^z are weight parameter matrices for the track or tag embedding, the previous hidden state, and the action embedding, respectively. b^{xz}, b^{hz}, and b^{az} are bias parameter vectors. The candidate hidden state is computed as follows

$$h'_t = \tanh(W x'_t + b^x + r_t \odot (U h_{t-1} + b^h) + Q a'_t + b^a) \tag{15}$$

with the reset gate r_t computed as

$$r_t = \sigma(W^r x'_t + b^{xr} + U^r h_{t-1} + b^{hr} + Q^r a'_t + b^{ar}) \tag{16}$$

With this extension, we make no explicit assumptions regarding the meaning and semantics of skips. Therefore, using the extended GRU definition, the hidden states are able to capture information about skips, which we hypothesise can help represent the intent of the user during a session.

3.3.2 History Extension. Working with the assumption that the preferences of a user evolve over time as shown both by Lamont and Webb [23] and Gupta et al. [12], we propose an extension to STABR which accounts for a user's prior sessions when predicting for their current session. Note that by actively accounting for prior sessions, we are, according to Quadrana et al. [31], working with

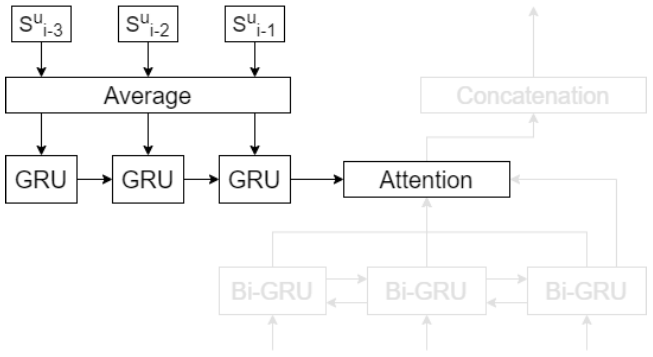


Figure 3: Overview of the history extension for STABR on one of the components when $x = 3$. Note that this is mirrored on the second component.

session-aware recommendation as opposed to session-based recommendation. However, we continue to focus just on predicting the next track in the currently active session. Specifically, we extend the attention layer of the existing STABR architecture such that it weighs the importance of tracks and tags based on both the last track played and the previous session(s) of the user. We extend both components—that is, the tracks and tags component seen in figure 1—such that user history is taken into account for tracks and tags, thus enabling the architecture to capture change in long-term preferences of both specific tracks and genres.

An overview of the history extension can be seen in figure 3. The extension takes as input the k previous session(s) of a user u, where k is a hyperparameter. An embedding of each track and tag in previous sessions are then obtained using the same embedding layers described in equations 1 and 2. In order to obtain a single vector representation of a session, we average the track and tag embedding vectors, such that

$$s^{u\prime}_{i-1,v} = \frac{1}{|s^u_{i-1}|} \sum_{j=1}^{|s^u_{i-1}|} v'_{i-1,j} \tag{17}$$

where $s^{u\prime}_{i-1,v}$ is the final vector representation of the tracks in session s^u_{i-1}, and $v'_{i-1,j}$ is the embedding of the j-th track in session s^u_{i-1}. For tags, equation 3 is first used to obtain averaged tag vectors for each track, then, similar to equation 17, a final vector representation of the tags is obtained, such that

$$s^{u\prime}_{i-1,g} = \frac{1}{|s^u_{i-1}|} \sum_{j=1}^{|s^u_{i-1}|} g'_{i-1,j} \tag{18}$$

where $s^{u\prime}_{i-1,g}$ is the final vector representation of the tags in session s^u_{i-1}, and $g'_{i-1,j}$ is the vector representation for the tags of the j-th track in session s^u_{i-1}. The track and tag representations of each previous session are then fed to GRU networks.

Note that we employ unidirectional GRU networks since we wish to maintain the order in which previous sessions occur such that the evolving preferences of the user is captured. We update the definition of α' and β' in equations 7 and 9 such that the final hidden state of the previous sessions is taken into account:

$$\alpha' = w^v \tanh(U^v P^v_j + b^{uv} + Q^v[\overrightarrow{h^v}; \overrightarrow{h^{s^u_v}}] + b^{qv}) + b^{wv} \tag{19}$$

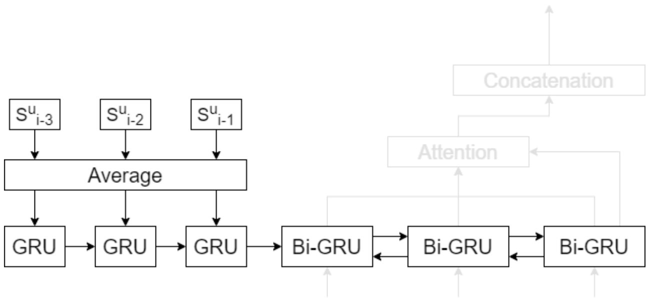


Figure 4: Overview of the ASLM-inspired history extension for STABR on one of the components when $x = 3$. Note that this is mirrored on the second component.

$$\beta' = w^g \tanh(U^g P^g_j + b^{ug} + Q^g[\overrightarrow{h^g}; \overrightarrow{h^{s^u_g}}] + b^{qg}) + b^{wg} \tag{20}$$

where $[\overrightarrow{h^v}; \overrightarrow{h^{s^u_v}}]$ is the concatenation of the final forward hidden state of the BiGRU network for tracks and the final hidden state for the tracks of the previous sessions. Using the extended attention layer, we enable the weights, α and β to also model the alignment of the hidden state representations of each time step in the active session with the tracks heard by the user in previous sessions. This, we hypothesise, helps STABR determine which tracks and tags in the current session are more important based on the evolution of the user's long-term preferences in terms of both tracks and tags.

Additionally, we consider a second variation of the history extension similar to the ASLM architecture presented by Huang et al. [19]. An overview of this variant can be seen in figure 4. In this variant, the track and tag representation presented in previous sections in equations 17 and 18 are also fed to GRU networks. However, the final hidden state is instead used as the initial state for the BiGRU networks used for track and tag embeddings. Thus, the two variants of the history extension consider the same information, but differ in how the information is exploited.

4 DATASETS

We employ two distinct datasets in order to (a) reproduce the results attained by Sachdeva et al.'s STABR architecture and explore our extensions described throughout section 3; and (b) investigate the merits of using complex architectures in terms of generalisability for music recommendation. Indeed, increasingly complex models can be troublesome if such complexity and associated increase in computational time is not accompanied by meaningful gains in performance as observed by Ludewig et al. [28]. Additional concerns are further raised by Dacrema et al. [10] following a systematic review of several recent, neural-based state-of-the-art recommendation algorithms and their ostensible reproducability: If a complex architecture cannot with reasonable effort be reproduced, its proclaimed progress gets called into question. We argue that two distinct datasets can help illuminate both the generalisability of a complex architecture, as well as the potential of our proposed extensions. In the realm of session-based music recommendation, we note that especially two datasets—and associated derivations—are frequently employed (e.g. as seen in [27, 31, 43]):

- *Lastfm-1K*, originally compiled by Òscar Celma [4], contains approximately 20 million listening events separated into listening sessions submitted by 992 distinct Last.fm users between 2005 and 2009.
- *30Music*, compiled by Turrin et al. [41] as part of RecSys 2015, contains approximately 5.6M tracks, approximately 31M listening events separated into listening sessions submitted by 45K distinct Last.fm users between 2014 and 2015.

We will be using modified versions of both sets. Specifically, since we (a) require tags as part of STABR; and (b) wish to account for skips, we necessarily need to augment both sets to include tags and track durations. Due to the structure of both datasets, we are, moreover, able to create histories by linking sessions of the same user—something which is a requirement for our history extensions. In the following two subsections, we detail the steps taken to pre-process and otherwise prepare our data as well as present general statistics of the sets. Note that while there is a small overlap between the tracks present in both sets, there is, to the best of our knowledge, no overlap between the users.

4.1 Lastfm-1K

In addition to being commonly employed, the Lastfm-1K dataset was also used by Sachdeva et al. [33] in their original paper presenting the STABR architecture. As such, we consider it a fitting choice to include this dataset as well. However, whilst Sachdeva et al. do rely on Last.fm to obtain their tags, they do not detail how exactly they do so, nor is their augmented data, to the best of our knowledge, publicly available. As such, in order to augment the Lastfm-1K dataset with tags, we primarily rely on the Last.fm API, but also conduct traditional crawling to account for instances in which the API fails to return all available tags.

In obtaining track durations, we similarly rely on the Last.fm API, but also reference Spotify's API[3] where applicable as to increase our coverage of tracks with duration. In the rare event where two different durations are available for the same track—usually a difference of no more than a couple of milliseconds—we compute the average. In order to compute skip values for tracks in a given session, we compare the duration of a track with the amount of seconds between the starting points of the current and the next track. Doing so, we obtain a value denoting how much of a track was played in a listening event before the next listening event was registered. Specifically, we consider a track to be skipped if less than 90% of a tracks duration has passed before the next listening event occurs. We choose a value of 90% rather than 100% to account for inaccuracies with timestamps of listening events and music playback features such as faded transitions (Spotify's default is five seconds, e.g.). Furthermore, we opt for 90% rather than a lower value as we are exclusively interested in knowing whether a user actively skipped a track, and not how much a track was played.

Similar to [33], we narrow our focus to consider a six month subset for each user in order to reduce the size of the dataset for our experiments. Specifically, for each user, we include listening events occurring from their first registered listening event and six months after that point in time. We discard any track with only a single play across the entire dataset. Sessions are constructed with a threshold of 120 minutes which is the same threshold used by Sachdeva et al. [33]. Thus, two consecutive listening events are considered to be in the same session if they occur within 120 minutes of each other. Similar to Sachdeva et al., we further discard any session with less than five listening events in total. Conclusively, due to a Tensorflow-specific issue, we also discard any session in which the first played track does not have any associated tags—something which affects approximately 0.1% of available sessions.

Description	Lastfm-1K	30Music
Number of users	836	10196
Number of sessions	68009	36471.67
Number of unique tracks	253621	209086.3
Number of unique tags	49301	52370.67
Avg. number of sessions per user	81.35	3.78
Avg. number of tracks per session	29.59	16.89
Avg. number of skips per session	4.88	0.1
Avg. number of plays per track	7.93	2.95
Avg. number of tags per track	4.47	3.48

Table 1: Descriptive statistics of the pre-processed Lastfm-1K dataset and the 30Music dataset averaged over the pre-processed slices.

Descriptive statistics of the final pre-processed dataset[4] can be seen in table 1. From the statistics, we see that in the average session of about 30 tracks, approximately every sixth track is skipped. Figure 5b shows the distribution of the percentage of tracks in sessions that have been skipped. The distribution shows that among sessions containing skipped tracks, (10%, 20%] of tracks are skipped most often followed by (0%, 10%]. In relatively few sessions are more than half the tracks skipped. Specifically, sessions with 50% of tracks or more being skipped account for 0.024% of all sessions. This may indicate a low level of "exploration" sessions where users attempt to discover new tracks, artists, or genres. However, it may also be a sign of an era in which music was less disposable given that the dataset was collected between 2005 and 2009, a time period before music streaming services like Spotify became widely popular. Further research is necessary to confirm this, however. At 22.2%, sessions with no skips account for a large percentage of the data.

Figure 5a shows the distribution of the number of sessions for users. The most common number of sessions is in the range 10-30, which, on average, translates to an active session from about every 18th day to every sixth day throughout the six months. However, there is also a sizeable portion, 35.3%, of users with 92 sessions or more—something which translates to active sessions ranging from every other day to multiple sessions per day. It is thus safe to conclude that most users in the Lastfm-1K dataset compile relatively large listening histories throughout six months.

Figure 5c shows the distribution of the percentage of tracks in sessions that are unique, i.e. have not been played more than once in a given session. This distribution shows that most sessions contain few to no repetitions as sessions with a percentage of unique tracks at 90-100% constitute 65.9% of the data. Sessions where half the tracks or more have been played more than once constitute a mere 0.092% of the data. This indicates that the data contains relatively few trivial cases in which a user exclusively listens to a small selection of tracks on repeat.

[3]https://developer.spotify.com/documentation/web-api/

[4]Processed data is available at https://github.com/x775/SW10/

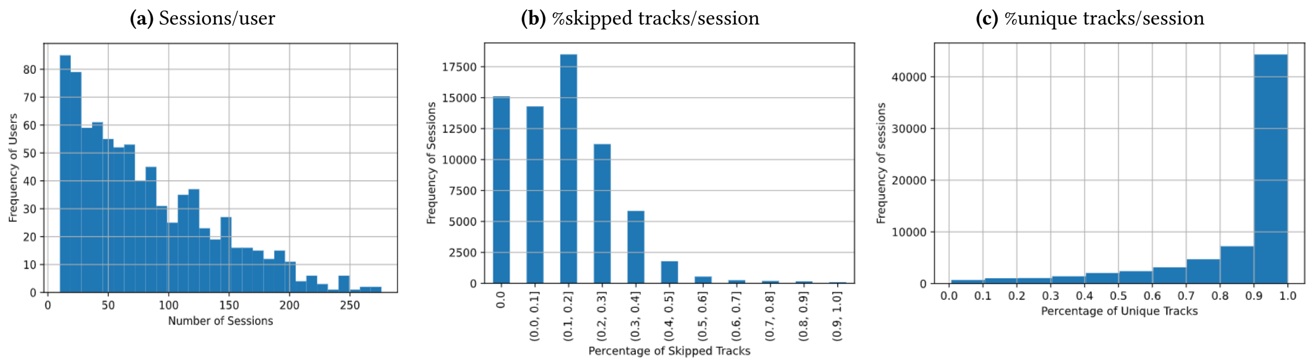


Figure 5: Selected graphs for the **Lastfm-1K** dataset. Figure 5a shows the distribution histogram of the number of sessions by frequency of users; figure 5b shows the distribution bar chart of the percentage of skipped tracks by frequency of sessions; and figure 5c shows the distribution histogram of percentage of unique tracks per frequency of sessions.

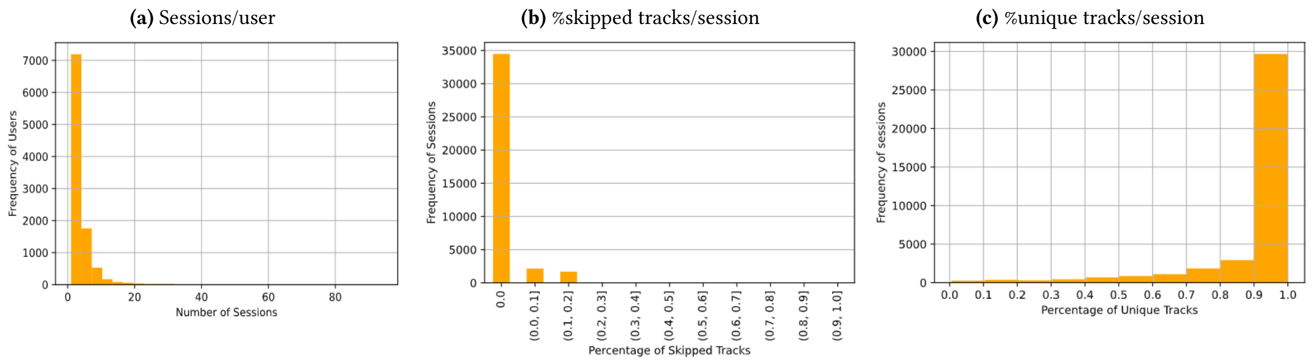


Figure 6: Selected graphs for the **30Music** dataset from a single slice; the remaining set of graphs are included in the appendix section A. Figure 6a shows the distribution histogram of the number of sessions by frequency of users; figure 6b shows the distribution bar chart of the percentage of skipped tracks by frequency of sessions; and figure 6b shows the distribution histogram of percentage of unique tracks per frequency of sessions.

4.2 30Music

In the case of 30Music, other work by Turrin et al. [40] happen to also include descriptive information such as tags and durations, and we are able to augment tracks using this. Since this data was collected prior to an update to the Last.fm API, it does not suffer from the issues requiring us to conduct additional crawling of the Lastfm-1K set. Since a subset of the 30Music set is used across multiple works by Ludewig et al. [27, 28], we opt for mirroring their pre-processing steps such that we can modify their *session-rec* framework for additional experiments. Specifically, this means that rather than using the full 30Music set, we will use three 90-day slices for training and three 5-day slices for testing similar to Ludewig et al. [27]. The reason for creating three slices as opposed to merely one is to get a more representative sample of the entire dataset. Additionally, as a result of replicating Ludewig et al.'s pre-processing steps, no unseen data will be present in the testing slices. This is in contrast to the Lastfm-1K set which has a high likelihood of containing unseen data in any split. We repeat the steps described in section 4.1 to compute skips for all 30Music slices. Descriptive statistics averaged across the final three pre-processed slices can also be seen in table 1. Most notably, the average number of skips per session is at 0.1, which is low considering that the data features 16.89 tracks per session on average. Figure 6b further highlights the low number of skips for the first slice of the 30Music data as sessions with no skips account for 89.77% of all sessions. Sessions with (0%, 20%] of tracks being skipped account for most sessions that contain skips. Specifically, in only 91 sessions have more than 20% of the tracks been skipped. Another significant difference from the Lastfm-1K dataset is the average number of sessions per user at 3.57. Given that the slices covers a period of 95 consecutive days when combining training and test data, the average user has an active session roughly every 27th day. Figure 6a also shows that most users in the first data slice have less than ten sessions, and that few users come close to a total of 20 sessions or more. Specifically, only 119 of the 9361 users have 20 sessions or more. Thus, most users in the 30Music dataset compile small listening histories compared to users in the Lastfm-1K dataset, even when accounting for the fact that each slice is only three months.

One aspect in which the 30Music dataset is similar to the Lastfm-1K dataset, is in terms of how many tracks in each session are unique. Figure 6c shows the distribution of the percentage of tracks in sessions that are unique for the first data slice. This figure reveals a distribution similar to the one obtained for the Lastfm-1K data shown in figure 5c. Thus, the 30Music dataset also seems to contain few trivial cases in which sessions contain a small selection of repeating tracks.

5 EVALUATION

We conduct experiments on both datasets described in sections 4.1 and 4.2. In order to comprehensively evaluate the potential of skips as a predictor for session-based music recommendation, we additionally consider a variation of the pre-processed Lastfm-1K data in which we discard all sessions not containing any skipped tracks such that the presence of skips is increased. We include this variation to test whether different distributions influence the predictive power of skips. We employ the following baselines:

- *POP*: The most popular tracks in the training set are recommended.
- *S-POP*: The most popular tracks in the current session are recommended with tiebreaks being resolved by using the most popular track in the training set.
- *SKNN*: A modified version of the Session-Based Collaborative Filtering (SSCF) model proposed by Park et al. [30]. Specifically, we compute
$$\hat{r}_{s,i} = \bar{r}_s + \frac{\sum_{v \in S(s)^k} sim(s,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in S(s)^k} sim(s,v)} \tag{21}$$
where $S(s)^k$ represents the k-th session neighbours for the current session s. $\bar{r}_s$ and $\bar{r}_v$ represent the average rating for sessions s and v under comparison. $r_{v,i}$ represents the frequency a track i is played in session v. $sim(s,v)$ represent the similarity between sessions s and v. In this specific implementation, we apply cosine similarity, though other similarity measures could also be used.
- *SKNN-SKIP*: We extend the above SKNN model to also account for skips. Specifically, in the event that a track has been skipped in a session, we do not increment its frequency. This approach means we exclusively consider skips as a negative signal.
- *SABR*: The feature-ignorant version of STABR as proposed by Sachdeva et al. [33]. We include this model to confirm the relevance of tags. Its implementation is identical to that of STABR as detailed in section 3.2, but only the component receiving tracks as input is used.
- *STABR*: The original, feature-aware STABR architecture as proposed by Sachdeva et al. [33] and as detailed throughout section 3.2.

As noted in section 4.2, no unseen data is present in the testing slices of the 30Music set. This property enables us to include a number of additional baselines for this set which assumes this property to be true. Specifically, we are including the following baselines in addition to those also employed with the Lastfm-1K dataset:

- *VSKNN*: Vector Multiplication Session-Based kNN as proposed by Ludewig et al. [28] is an extension to the traditional SKNN which places increased emphasis on the more recent events of a session when computing similarities. Specifically, the weights of elements are determined by a linear decay function which depends on the position of the element within the specific session—effectively ensuring elements appearing earlier in the session get a lower weight.
- *FPMC*: A matrix factorisation and Markov Chain-based model proposed by Rendle et al. [32] and introduced in section 2.1.
- *FOSSIL*: An alternative matrix factorisation and Markov Chain based model proposed by He and McAuley [13]. The overarching goal of FOSSIL is to address data sparsity issues sometimes hampering the original FPMC model.
- *GRU4REC*: The original implementation as proposed by Hidasi et al. [15] and otherwise introduced in section 2.1.
- *GRU4REC2*: The same as GRU4REC, but with an updated loss function as proposed by Hidasi et al. [14] and mentioned in section 2.1.

In addition to the abovementioned baselines, we evaluate our extensions on both datasets:

- *STABR-SKIPS*: The extension proposed in section 3.3.1.
- *STABR-HIST*: The extension proposed in section 3.3.2.
- *STABR-HIST-II*: The alternative approach to including history inspired by the ASLM architecture by Huang et al. [19] as proposed in section 3.3.2.

5.1 Experimental Setup

We employ categorical cross entropy as the loss function for training the STABR architecture and our proposed extensions such that the loss of a training session is obtained as:

$$-\sum_{i=0}^{|V|} y_i \log(\hat{y}_i) \tag{22}$$

where y_i is a one-hot vector representing the target track, and $\hat{y}_i$ is the probability distribution obtained from the softmax activation in the last layer of the STABR architecture (see figure 1). Similar to Sachdeva et al. [33], we employ Adagrad by Duchi et al. [11] as our optimiser algorithm. In short, Adagrad automatically scales parameter-specific learning rates according to the frequency of which a given parameter is updated during training. This avoids the issues of setting the learning rate too high, too low, or merely fitting to a single dimension. In separating our data for training and testing, we once again refer to Sachdeva et al. and use the first 70% of each users' sessions in order of occurrence as training data for the Lastfm-1K dataset. The next 10% is used as validation data, and the remaining 20% is used as testing data. In preparing the 30Music dataset, we follow the procedure outlined in section 4.2 such that three 90-day splits are used as training data, and three 5-day (corresponding to the subsequent five days following the 90th day) splits are used as test data. Similar to Ludewig et al. [28], we also create a validation set for the 30Music dataset. We do this by randomly sampling 10% of the training data. We follow the approach of Li et al. [25] and implement sequence splitting such that for the input session $\{v_1, v_2, ..., v_{i-1}, v_i\}$, we generate sub-sessions: $\{v_1, v_2\}$, $\{v_1, v_2, v_3\}$, ..., $\{v_1, v_2, ..., v_{i-1}\}$. We complete this step for both datasets. The last track of each sub-session is the target track.

In tuning our hyperparameters for STABR-based models, we employ the Hyperband algorithm as proposed by Li et al. [26] due to (a) the practicality of it being available through the Keras-tuner library[5]; and (b) the fact that it provides faster optimisation than conventional tuning algorithms by exploiting random sampling and early stopping. We complete between 40 and 70 trials depending

[5]https://github.com/keras-team/keras-tuner

Lastfm-1K Hyperparameters					
	SABR	**STABR**	**STABR-SKIPS**	**STABR-HIST**	**STABR-HIST-II**
Batch Size	32	32	32	32	32
Hidden Size	32	96	96	96	96
Learning Rate*	0.15	0.2	0.2	0.2	0.2
Tag Emb. Size	*N/A*	100	100	100	100
Track Emb. Size	50	50	50	50	50
Dense Layer Size	75	50	75	50	50
Dropout	0.1	0.1	0.1	0.1	0.1
Skip Emb. Size	*N/A*	*N/A*	15	*N/A*	*N/A*
History Size	*N/A*	*N/A*	*N/A*	10	10
Tag Hist. HS**	*N/A*	*N/A*	*N/A*	64	96
Track Hist. HS**	*N/A*	*N/A*	*N/A*	64	96

**initial learning rate, **HS = Hidden Size*

Table 2: Hyperparameters for STABR-based Lastfm-1K models.

30Music Hyperparameters					
	SABR	**STABR**	**STABR-SKIPS**	**STABR-HIST**	**STABR-HIST-II**
Batch Size	32	32	32	32	32
Hidden Size	64	64	96	64	64
Learning Rate*	0.15	0.15	0.2	0.15	0.15
Tag Emb. Size	*N/A*	100	100	100	100
Track Emb. Size	75	75	50	75	75
Dense Layer Size	25	25	75	25	25
Dropout	0.1	0.1	0.1	0.1	0.1
Skip Emb. Size	*N/A*	*N/A*	15	*N/A*	*N/A*
History Size	*N/A*	*N/A*	*N/A*	10	10
Tag Hist. HS**	*N/A*	*N/A*	*N/A*	64	64
Track Hist. HS**	*N/A*	*N/A*	*N/A*	64	64

**initial learning rate, **HS = Hidden Size*

Table 3: Hyperparameters for STABR-based 30Music models.

on the model before selecting our parameters. The final hyperparameters for STABR-based models can be seen in table 2 for the Lastfm-1K dataset, and table 3 for the 30Music dataset. Additional hyperparameters for the 30Music dataset baselines are included in appendix B. Note that while the learning rate may appear high, it is the initial learning rate and will automatically be tweaked as our models progress in training due to us employing Adagrad.

Similar to Sachdeva et al. [33], we also employ dropout [36] for the first dense layer in the STABR architecture and all the extended variants. This helps reduce the risk of overfitting by randomly ignoring neurons in the dense layer during training such that the effect of co-adaptations between individual neurons is reduced.

We train all neural-based models using a single NVIDIA Tesla V100 with TensorFlow version 2.1.0 for STABR-based models and Theano 1.0.3 for GRU4REC and GRU4REC2. We train all kNN- and factorisation-based models on clusters with 3.1GHz Intel Xeon SkyLake CPUs. In the case of SKNN and SKNN-SKIPS for the Lastfm-1K dataset specifically, we use 100 and 125 neighbours, respectively. Indeed, we find that the performance gain obtained by increasing the amount of neighbours stagnates around these numbers. This is illustrated in figure 10 in appendix C. We employ two metrics for evaluating our baselines and proposed extensions:

- *Hit Rate (HR)*: A measure of how frequently the target item is found in the top k recommendations of a model. The metric is calculated as $HR = \frac{hits}{total}$, where $hits$ is the number of test examples wherein the target item is found in the top k recommended items, and $total$ is the total number of test examples. This is equivalent to the recall metric known from information retrieval [49].
- *Mean Reciprocal Rank (MRR)*: A measure for evaluating a model's ability to rank the target item highly among top k recommendations in terms of assigned probability [9]. Specifically, the MRR is calculated as follows:

$$MRR = \frac{1}{total} \sum_{i=1}^{total} \frac{1}{rank_i} \tag{23}$$

where $total$ is the total number of test examples, and $rank_i$ is the rank of the target item for test example i. As an example, if a target item is assigned the second highest probability among the top k recommendations, the rank of the target item is 2 and the reciprocal rank becomes $\frac{1}{2}$. On the other hand, if the target item is not among the top k recommendations, the reciprocal rank is 0.

Note that some papers, e.g. [10] and [27], employ the Normalized Discounted Cumulative Gain (NDCG) metric which is similar to MRR in nature. Since we are not dealing with a ranking akin to search results and similar, but rather whether a track was played or not, and thus have no relevancy score for any track, we consider MRR better suited for our use-case.

5.2 Research Questions

Based on our hypotheses and extensions to the STABR architecture expounded throughout section 3, we consider the following research questions for evaluation:

RQ1 Does the STABR architecture and session-based k-nearest neighbours benefit from being aware of skips during a session in terms of *HR* and *MRR*?

RQ2 Does the STABR architecture benefit from being aware of a user's previous sessions when recommending for their currently active session in terms of *HR* and *MRR*?

RQ3 Does the STABR architecture and our extensions consistently outperform simpler approaches to session-based recommendation such as k-nearest neighbours?

Lastfm-1K

	HR@					MRR@				
	10	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
POP	0.0007	0.0018	0.0033	0.0047	0.0057	0.0003	0.0003	0.0004	0.0004	0.0005
S-POP	0.1269	0.1749	0.1966	0.2098	0.2169	0.0586	0.0620	0.0629	0.0632	0.0634
SKNN	0.1907	0.2694	0.3066	0.3332	**0.3534**	0.0639	0.0695	0.0710	0.0717	0.0722
SKNN-SKIPS	0.1777	0.2473	0.2832	0.3088	0.3280	0.0602	0.0652	0.0666	0.0673	0.0678
SABR	0.2847	0.3042	0.3159	0.3247	0.3316	0.2336	0.2350	0.2354	0.2357	0.2358
STABR	**0.2976**	0.3197	0.3328	0.3427	0.3500	**0.2402**	**0.2417**	**0.2422**	**0.2425**	**0.2427**
STABR-SKIPS	0.2891	0.3091	0.3205	0.3291	0.3355	0.2363	0.2377	0.2381	0.2384	0.2385
STABR-HIST	0.2944	0.3182	0.3318	0.3416	0.3493	0.2355	0.2372	0.2377	0.2380	0.2382
STABR-HIST-II	0.2970	**0.3201**	**0.3337**	**0.3439**	0.3517	0.2378	0.2385	0.2399	0.2402	0.2404

Table 4: All results from experiments conducted on the Lastfm-1K dataset. The best result for each metric is boldfaced.

30Music

	HR@					MRR@				
	10	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
POP	0.0027	0.0047	0.0067	0.0084	0.0115	0.0006	0.0008	0.0008	0.0009	0.0010
S-POP	0.0987	0.1185	0.1283	0.1320	0.1358	0.0512	0.0526	0.0530	0.0531	0.0532
SKNN	0.2517	0.3466	0.3942	0.4244	**0.4444**	0.0855	0.0922	0.0934	0.0943	0.0947
VSKNN	0.3057	**0.3815**	**0.4118**	**0.4301**	0.4440	0.1120	0.1174	0.1186	0.1192	0.1195
FMPC	0.0729	0.0819	0.0877	0.0912	0.0945	0.0555	0.0561	0.0564	0.0565	0.0565
FOSSIL	0.0295	0.0408	0.0497	0.0556	0.0604	0.0123	0.0132	0.0135	0.0137	0.0138
GRU4REC	0.3007	0.3246	0.3364	0.3449	0.3511	0.2310	0.2326	0.2331	0.2333	0.2335
GRU4REC2	**0.3322**	0.3533	0.3626	0.3685	0.3734	**0.2457**	**0.2472**	**0.2475**	**0.2477**	**0.2478**
SABR	0.2097	0.2233	0.2313	0.2381	0.2420	0.1775	0.1785	0.1788	0.1790	0.1791
STABR	0.2024	0.2163	0.2255	0.2308	0.2364	0.1702	0.1712	0.1715	0.1717	0.1719
STABR-SKIPS	0.1943	0.2082	0.2171	0.2239	0.2292	0.1570	0.1580	0.1583	0.1586	0.1587
STABR-HIST	0.1818	0.1935	0.2041	0.2100	0.2155	0.1483	0.1493	0.1497	0.1498	0.1499
STABR-HIST-II	0.1823	0.2016	0.2119	0.2173	0.2234	0.1439	0.1450	0.1458	0.1459	0.1463

Table 5: All results averaged from experiments conducted on the 30Music dataset. The best result for each metric is boldfaced.

Lastfm-1K-Skips

	HR@					MRR@				
	10	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
SKNN	0.1778	0.2536	0.2898	0.3164	**0.3371**	0.0594	0.0647	0.0662	0.0670	0.0674
SKNN-SKIPS	0.1630	0.2275	0.2621	0.2870	0.3061	0.0552	0.0598	0.0611	0.0619	0.0624
STABR	**0.2846**	**0.3066**	**0.3194**	**0.3287**	0.3361	**0.2303**	**0.2318**	**0.2324**	**0.2326**	**0.2328**
STABR-SKIPS	0.2720	0.2964	0.3107	0.3211	0.3292	0.2117	0.2134	0.2139	0.2142	0.2144

Table 6: All results from experiments conducted on the Lastfm-1K dataset where all sessions have skips. The best result for each metric is boldfaced.

5.3 Results and Discussion

In the following sections, we present our results and discuss them in the context of each of the three research questions presented in section 5.2. All results discussed throughout this section are available in tables 4, 5, and 6.

5.3.1 RQ1. As noted in section 3.3.1, skips constitute implicit feedback provided by the user during a session. Our hypothesis is that by actively accounting for skips, we are able to better capture the consumption motivations—i.e. intent—and thus the short-term preferences of a user during a given session. In order for this hypothesis to hold, we should thus expect our STABR-SKIPS and SKNN-SKIPS extensions to attain increased performance compared to the skip-ignorant STABR and SKNN architectures, respectively.

A fundamental requirement for STABR-SKIPS and SKNN-SKIPS to have any chance of confirming our hypothesis is necessarily that skips are present in the data. As such, considering the substantially larger number of average skips per session in the Lastfm-1K dataset (4.88) compared to that of the 30Music dataset (0.1), we should expect the greatest difference to occur with the Lastfm-1K set.

What quickly becomes apparent in table 4 is how the STABR-SKIPS extension is inferior to both the original STABR as well as our history-based extensions across all *HR@k* values. Additionally, while we note a small uptick in the *MRR@*10 performance compared to STABR-HIST—0.2363 versus 0.2355—the performance of STABR-SKIPS remains inferior to that of the original STABR and STABR-HIST-II across all *MRR@k* values. It performs approximately on par with STABR-HIST on *MRR@k* with $k \geq 20$. On the other hand, STABR-SKIPS does hold a narrow lead against the feature-ignorant SABR across all *HR@k* and *MRR@k* values. This, however, is more likely due to it also accounting for tags than it accounting for skips. In fact, that STABR-SKIPS scores very similar to SABR in the first place suggests that any gain attained by including tags is nullified by accounting for skips.

We further observe a similar pattern with our SKNN-SKIPS extension: Compared to the standard SKNN, the extension performs poorly across all *HR@k* and *MRR@k* values. It is in this regard important to note how the SKNN-SKIPS extension exclusively considers skips as negative signals, whereas the STABR-SKIPS extension can consider skips as either positive or negative signals. Interestingly, neither approach seems effective in improving performance compared to their skip-ignorant counterparts. In other words, neither the STABR nor the SKNN architecture seems to immediately benefit from being aware of skips during a session in the Lastfm-1K dataset. What is more, given the reduced performance, there is a general argument to be made that accounting for skips introduces more noise than valuable signals indicative of the user's short-term preferences in the Lastfm-1K dataset. In pursuit of affirming this, and in an attempt to reduce the risk of inadvertently introducing unnecessary noise, we devise an alternative version of the STABR-SKIPS extension in which we exclusively account for skips in the tracks-component as opposed to both components. However, this extension only further diminishes performance.

On the 30Music dataset presented in table 5, our STABR-SKIPS extension, similarly, does not outperform STABR nor any of our other extensions. Compared to the Lastfm-1K dataset, however, the presence of skips in this dataset is significantly lower. As such, the results attained on the 30Music dataset are largely expected as we by any account are introducing more noise than signal. What is worth noting, however, is how STABR-SKIPS outperforms our two history extensions by a significant margin—something which it did not on the Lastfm-1K set. We will discuss this in section 5.3.2.

While the results presented in tables 4 and 5 are not exactly promising, we are hesitant to definitively dismiss the value of skips in general: It might be that neither dataset contains sufficient skips to derive any value. Indeed, and as speculated in section 4.1, newer data could reflect changed consumption behaviours favouring more skips as a result of increased disposability per the nature of streaming. As such, and as briefly noted in section 5, we devise an alternative version of the Lastfm-1K dataset consisting only of sessions containing skips. Compared to the complete dataset with an average number of skips per session of 4.88, the Lastfm-1K-Skips set has approximately 29% more skips with an average of 6.29 per session.

Note that since we are exclusively interested in determining the efficacy of including implicit feedback in the form of skips, we only run the SKNN and STABR baselines alongside their skips extensions on this set. Additionally, we do not repeat this experiment with the 30Music dataset given its already low number of skips. The results from this dataset are presented in table 6. Once again, neither extension manages to outperform its skip-ignorant counterpart. In fact, the relative difference in performance across all *HR@k* and *MRR@k* values between STABR-SKIPS and STABR as well as SKNN-SKIPS and SKNN is slightly greater in this set than in the normal Lastfm-1K set. Correspondingly, these results further support the notion that skips not merely constitute poor signals, but in fact introduce noise when accounted for in data similar to the Lastfm-1K and 30Music sets. As such, in answering **RQ1**, no additional performance for either architecture is obtained by actively accounting for skips in currently available data.

5.3.2 RQ2. Our hypothesis is that accounting for long-term preferences yields a more accurate user representation from which more accurate predictions can be computed. Indeed, where the original STABR architecture considers each session independently—and thus exclusively accounts for short-term preferences—the work presented throughout section 2.2 (e.g. Chen et al. [5] or Tang et al. [39]) suggests that preferences not only evolve over time, but that value can be derived from long-term preferences in general. It is in this regard worth accentuating how the 30Music dataset contains substantially less sessions per user (3.78) compared to Lastfm-1K (81.35) to the point where a significant portion of users have next-to-no history. As such, and similar to **RQ1**, we should expect the greatest difference to occur with the Lastfm-1K set.

As seen in table 4, the STABR-HIST-II extension consistently outperforms the STABR-HIST extension across all *HR@k* and *MRR@k* values on the Lastfm-1K set. Recall, where STABR-HIST feeds a representation of the user's prior sessions directly to the attention layer, STABR-HIST-II uses the same representation as the initial hidden state for the BiGRU networks. The results attained here suggests that this latter approach is more beneficial. The representation of the user's past k sessions thus proves a better initial hidden state, ultimately producing a more accurate output hidden state representation at the end of the BiGRU networks.

What is more, STABR-HIST-II narrowly outperforms all other baselines on *HR*@20, *HR*@30, and *HR*@40, as well as outperforms all but SKNN on *HR*@50. STABR-HIST-II does, however, fall short of STABR across all *MRR@k* values as well as *HR*@10. At face value, this suggests that in terms of hit rate, accounting for prior sessions in predicting the next-track in the current session remains a promising endeavour. We conduct a one-tailed sign test [34] in order to compare STABR-HIST-II with STABR. We ignore samples where the two approaches ties; that is, when both are either correct or incorrect. The test produces the following p-values:

HR@20	$p = 0.1928$
HR@30	$p = 0.0183$
HR@40	$p = 0.0032$
HR@50	$p = 0.0002$

We observe that while there is a narrow difference for the *HR*@20 score in favour of STABR-HIST-II, it cannot be considered significant at significance level $\alpha = 0.05$. However, for *HR@k* with $k \geq 30$, STABR-HIST-II can be considered to significantly outperform STABR. In the case of *HR*@50, while SKNN scores the best overall, STABR-HIST-II significantly outperforms STABR. The fact that the higher k, the better performance of STABR-HIST-II suggests that the tracks STABR misses can be identified through the prior sessions of the given user. On the other hand, in regards to *MRR* for both STABR-HIST and STABR-HIST-II, we observe a drop across all *MRR@k* values compared to STABR. This is interesting as it suggests that while long-term dependencies appear to help increase the hit rate, it does not necessarily help output the correct answer sooner. In fact, doing so appears to gently skew the order of tracks in the wrong direction in terms of performance.

On the 30Music set, however, neither STABR-HIST nor STABR-HIST-II performs very well. In fact, they both perform below STABR as well as the feature-ignorant SABR across all *HR@k* and *MRR@k* values. We speculate that this is the result of the substantially lower average amount of sessions per user: If no, or only a few, sessions constitute a user profile, we risk either introducing nothing at all or just noise. Additionally, with only 3.78 sessions per user during a 90-day period, it might also be that the few sessions present are too far apart to have any interconnected relationships in regards to preference. We note that the relative difference in performance of the two extensions is a lot smaller (e.g. 0.0005 between STABR-HIST and STABR-HIST-II for *HR*@10) on this dataset compared to the Lastfm-1K set. We speculate that since HIST-II generally outperforms HIST on the Lastfm-1K set, a lack of user histories especially hampers this extension.

In other words: If sufficiently rich user history representations cannot be established, attempting to include such representations when predicting the next-track for the current session is akin to introducing noise. This is further supported by the fact that STABR-SKIPS outperforms both history extensions on the 30Music dataset despite scoring below both on the Lastfm-1K set across all *HR@k* values. This, however, is not a problem we can immediately solve as it relates to the infamous cold-start problem.

On the other hand, seeing how STABR-HIST-II generally keeps up with STABR on the Lastfm-1K set, and on some occasions outperforms it, we argue that there is reason to conduct further experimentation given adequate data. However, the fact that actively accounting for long-term preferences through prior sessions does not produce significantly better results at lower *k* hit rates, and at no *MRR@k* values, suggests a couple of things: Specifically, we might not be accounting for long-term preferences optimally, and/or short-term preferences could be the most important factor in session-based music recommendation for the data used in our experiments.

As such, in answering **RQ2**, the STABR architecture does not immediately seem to significantly benefit from being aware of a user's previous session at *HR@k* with $k \leq 20$ on the Lastfm-1K set. However, at higher *k*-values, STABR-HIST-II significantly outperforms STABR at significance level $\alpha = 0.05$. While we do not see the same results reproduced on the 30Music set, we argue that this is due to the low amount of average sessions per user. As such, we are hesitant to dismiss the value offered by long-range dependencies in general due to the promising performances at higher *k* hit rates on the Lastfm-1K dataset with STABR-HIST-II. These results ultimately suggests that value can be derived from histories, but that knowing how much to include and, perhaps more importantly, how to include it is the crucial question warranting further research.

5.3.3 RQ3. Given that the efficacy and perceived progress of recent state-of-the-art, complex neural approaches in session-based recommendation has been brought into question by both Ludewig et al. [28] and Dacrema et al. [10], we consider an evaluation of the performance of STABR, our proposed extensions, and simpler approaches vital for future research. Additionally, since STABR is originally only evaluated on a single dataset, evaluating the architecture on a second, distinctly different dataset, can further help illuminate the overall generalisability of the architecture as well as delineate circumstances under which complex architectures are justifiably. Indeed, as illustrated both in table 1, as well as figures 5 and 6, there are distinct differences between the Lastfm-1K and 30Music datasets also reflected in the performance of our baselines.

On the Lastfm-1K dataset, SABR, STABR, and our extensions all outperform simpler approaches like POP and S-POP across all *HR@k* and *MRR@k* values. They similarly outperform SKNN on *HR*@10, *HR*@20, and *HR*@30. However, for higher *HR@k* values, the SKNN architecture proves competitive, and even manages to outperform all other approaches for *HR*@50. Worth noting is that while SKNN-SKIPS also increases its performance on higher *HR@k* values proportionally, it never matches that of the standard SKNN. Indeed, a similar pattern is observed on the Lastfm-1K-Skips set in which the standard SKNN model also produces the best results for *HR*@50. These results reinforce the concerns raised by Ludewig et al. and Dacrema et al. as they suggest that under certain circumstances, simpler approaches are not merely competitive, they are sometimes superior to even complex neural approaches.

Our results attained on the 30Music dataset further support this seeing how the highest scores for all values of *HR@k*, excluding @10, is attained by either the SKNN or the VSKNN model. Worth noting is how neither factorisation-based model (FMPC and FOSSIL) performs very well. These results match those attained by Ludewig et al. [27, 28], and suggests that KNN and neural-based models are likely best suited for session-based music recommendation.

While SKNN and its variants manages to produce good results in terms of hit rate, they are not competitive approaches in terms of MRR. Indeed, as seen across tables 4, 5, and 6, SKNN, VSKNN, and SKNN-SKIPS all perform well below the neural approaches on *MRR@k* for all values. Thus, we observe that while SKNN performs well in terms of hitting the target track, when it comes to ranking the target track highly, it is consistently outperformed by neural approaches. This suggests that in a practical setting, the optimal choice between a simpler approach such as SKNN, and more complex neural approaches such as STABR or GRU4REC, is largely dependent on the context in which the recommender system will be used. If, for instance, the goal is to present a user with a list of tracks that they may enjoy based on tracks in their current session, SKNN is very capable. However, if the goal instead is automatic continuation of the current session—in which only the top recommended track matters—neural approaches may be better suited per their ability to more accurately rank the correct target track.

Among neural approaches, table 5 reveals that SABR, STABR, and all extended STABR variants are outperformed by both GRU4REC variants across all metrics on the 30Music dataset. In terms of *MRR*, GRU4REC2 proves the best performing approach. While GRU4REC and STABR are similar approaches in the sense that they both employ GRU networks, there are two major differences between the architectures: GRU4REC does not consider item features such as tags, nor does it employ an attention mechanism similar to STABR. The fact that the GRU4REC architecture performs better than STABR and our extensions on the 30Music dataset could suggest that

- tags are not good predictors for the 30Music dataset, and may in fact add more noise than signal to the model; and
- employing an attention mechanism is not immediately beneficial for the 30Music dataset.

The first claim is supported by the fact that SABR, the feature-ignorant version of STABR, performs better than STABR and all its

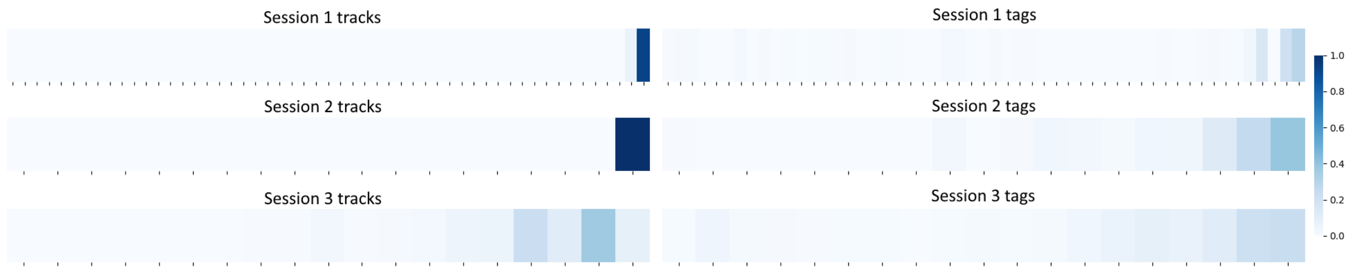


Figure 7: Heatmaps of the tracks and tags attention weights for three randomly sampled sessions from the **30Music** dataset. Darker shades of blue represents a higher weight assigned by the attention mechanism.

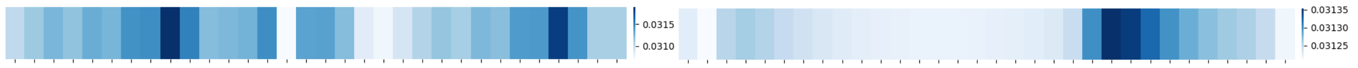


Figure 8: Heatmaps of the tracks (left) and tags (right) attention weights for one randomly sampled session from the **Lastfm-1K** dataset. Darker shades of blue represents a higher weight assigned by the attention mechanism.

extended variants across all metrics. This is most likely explained by a lower number of tags per track in the 30Music dataset compared to the Lastfm-1K dataset as established in table 1. Indeed, comparing the amount of tags between the two datasets reveals a significant difference: 1.25% of all tracks in the Lastfm-1K have no associated tags, while for the 30Music data, tracks with no tags constitute 22.29% of all tracks averaged over the three slices. Thus, with relatively many tracks without any tags for STABR to exploit, the architecture's ability to establish meaningful patterns of genre descriptors in 30Music sessions is hampered.

In order to investigate the second claim of whether an attention mechanism is beneficial for the 30Music dataset, we visualise the item weights for three randomly sampled sessions in the 30Music test data. Heatmaps of the weights applied by the attention mechanism for both tracks (left) and tags (right) can be seen in figure 7. Overall, these heatmaps show that the attention weights for all three sampled sessions are heavily focused towards the end—i.e. the most recent—of each session for both tracks of tags. This is especially apparent for the first session in which the last track is assigned close to 100% of the attention. Similarly, for the second session, the last two tracks are assigned close to 100% of the attention. In session three, on the other hand, the attention is somewhat more distributed with the fourth and second last track being assigned the highest weights. Similarly, for tags in all three sessions, the attention is also spread, but still primarily focused around the tail of each session. In sessions such as these, we argue that the attention mechanism might not have been an advantageous component. Indeed, traditional GRUs might have been able to produce similar results. Of course, this argument assumes that there really are no long-range dependencies present in the sampled sessions, and that our attention mechanism has not failed to identify them.

However, while the sampled sessions from the 30Music set do not immediately suggest the presence of long-range dependencies, the story is different on the Lastfm-1K set: The visualised attention weights of tracks and tags from a single, randomly sampled session from the Lastfm-1K dataset is seen in figure 8. Note that the scale in this figure is changed compared to figure 7, and that they thus cannot be directly compared. However, it is conspicuous that the attention weight values are more evenly distributed across the session relative to any of the 30Music sessions. This is evident from the fact that the highest value observed for both tracks and tags is only slightly higher than 0.03. The biggest weights, furthermore, do not seem to be centered around the tail of the session as they are for the 30Music sessions. Obviously, any conclusion regarding the effectiveness of attention cannot be drawn due to the small sample size. However, these results, nonetheless, suggest that an attention mechanism could be less useful for the 30Music set, and similar data, if the patterns in figure 7 are prevalent throughout. On the other hand, and with the same reservations regarding sample size as above, the attention mechanism quite clearly captures long-range dependencies on the Lastfm-1K set.

As such, in answering **RQ3** asking whether the STABR architecture and our extensions consistently outperform simpler approaches, the answer is twofold. Specifically, while STABR and our extensions consistently outperform simpler approaches such as SKNN and VSKNN in terms of *MRR@k* for all values, no STABR-based model consistently outperforms SKNN or VSKNN in terms of hit rate. This applies to both datasets. Notably, on the 30Music dataset, the STABR architecture is also beaten by the GRU4REC architecture across all *HR@k* and *MRR@k* values. However, since the GRU4REC architecture assumes no unseen data to be present in the testing set, this limitation naturally reduces GRU4REC's overall generalisability. This, arguably, gives the STABR architecture an edge in some datasets. However, sound performance from STABR appears to also require the presence of tags—something which similarly limits the type of data in which STABR can be effective. On the other hand, even without many tags and ignoring GRU4REC, the STABR architecture still outperforms all other baselines on *MRR@k* on both datasets.

6 CONCLUSION AND FUTURE WORK

In this paper, we implement the state-of-the-art session-based music recommender, STABR, and explore extensions to the architecture seeking to better accommodate (a) short-term preferences by incorporating implicit feedback during sessions; and (b) long-term preferences by accounting for prior sessions.

Specifically, we implement custom, contextual GRU networks in order to enable the STABR architecture to capture implicit feedback in the form of skips during the current session. Additionally, we extend the STABR architecture with a third, GRU-based component which considers the tracks and tags present in a user's prior sessions to accommodate long-term preferences. We consider two distinct approaches for exploiting this representation of long-term preferences: As information used directly in the attention mechanism, and as the initial hidden state for the two BiGRU networks in the main STABR architecture.

The results of our experiments conducted on two real-world datasets—Lastfm-1K and 30Music—indicate that skips cannot be considered worthwhile predictors for the next track of a session. Indeed, they seem to consistently introduce more noise than signal. In accommodating long-term preferences, we find that feeding a representation of prior sessions directly to the attention mechanism adds noise. However, using the same long-term preferences representation as the initial hidden states in STABR's BiGRU networks allows the extended architecture to beat the original STABR and all baselines in terms of hit rate—specifically $HR@20$, $HR@30$, and $HR@40$—on the Lastfm-1K dataset. STABR-HIST-II, moreover, outperforms STABR on $HR@50$. Since the results for $HR@30$, $HR@40$, and $HR@50$ are significant based on a one-tailed sign test, we argue that they warrant further research in order to fully gauge the efficacy of fusing long-term preferences with short-term preferences in session-based music recommendation.

Another point of focus throughout our paper has been the overall efficacy of complex neural-based models. In comparing STABR and our extensions to simpler non-neural approaches such as session-based k-nearest neighbours, we find that STABR and other neural approaches are not consistently the best option. Indeed, our results show that while STABR and other neural approaches remain superior in terms of mean reciprocal rank, SKNN and VKSNN proves competitive—and sometimes even superior—in terms of hit rate. This, we argue, only further accentuates the importance of not merely understanding your data, but also recognising which problem your session-based recommender system ought to solve.

In future work, we would like to conduct further investigation in the use of long-term preferences. While we have considered two extensions for employing the historical sessions of a user, they are similar in the sense that they use a static number of previous sessions which are always the sessions which occurred immediately before the active one. One direction we consider interesting is using a dynamic number of previous sessions determined by the individual user. This may lead to better representations of long-term preferences as they can then be customised based on the preferences of each user. Another direction would be to not only consider how many previous sessions are used but also the selection of those sessions. Merely using the immediate previous sessions may be too naive. A better approach may be to select sessions based on a similarity measure for the currently active sessions. Such a measure could account for both tags and other item-specific features such as audio features like tempo and valence.

While our results do not support skips as being useful signals, we remain hesitant to dismiss their potential entirely. In this paper, we have considered skips as binary features—"*skipped*" and "*not skipped*". Thus, one direction for further research is considering how much of a track was played before being skipped. Intuitively, it is sensible that skipping a track five seconds after it started playing is different from skipping a track after more than half of it has been played. Such information may hold value in the recommendation task.

7 ACKNOWLEDGEMENTS

We would like to thank Peter Dolog for his supervision and thoughtful feedback throughout this project.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[2] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166. https://doi.org/10.1109/72.279181

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109 – 132. https://doi.org/10.1016/j.knosys.2013.03.012

[4] O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail.* Springer.

[5] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. A Dynamic Co-Attention Network for Session-Based Recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1461–1470. https://doi.org/10.1145/3357384.3357964

[6] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR* abs/1409.1259 (2014). arXiv:1409.1259 http://arxiv.org/abs/1409.1259

[7] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[8] The Nielsen Company. 2017. Entertainment: U.S Music 360 - 2017 Highlights. https://www.nielsen.com/us/en/insights/report/2017/music-360-2017-highlights/. (2017). Accessed 2020-04-30.

[9] Nick Craswell. 2009. *Mean Reciprocal Rank.* Springer US, Boston, MA, 1703–1703. https://doi.org/10.1007/978-0-387-39940-9_488

[10] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 101–109. https://doi.org/10.1145/3298689.3347058

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12, 61 (2011), 2121–2159. http://jmlr.org/papers/v12/duchi11a.html

[12] Kartik Gupta, Noveen Sachdeva, and Vikram Pudi. 2018. Explicit Modelling of the Implicit Short Term User Preferences for Music Recommendation. In *Advances in Information Retrieval*, Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury (Eds.). Springer International Publishing, Cham, 333–344.

[13] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. (09 2016).

[14] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 843–852. https://doi.org/10.1145/3269206.3271761

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.06939

[16] Sepp Hochreiter. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 06, 02 (1998), 107–116. https://doi.org/10.1142/S0218488598000094

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[18] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models. 241–244. https://doi.org/10.1145/2792838.2799684

[19] Ruo Huang, Shelby McIntyre, Meina Song, Heihong E, and Zhonghong Ou. 2018. An Attention-Based Recommender System to Predict Contextual Intent Based on Choice Histories across and within Sessions. *Applied Sciences* 8, 12 (2018).

https://doi.org/10.3390/app8122426

[20] Dietmar Jannach. 2018. Keynote: Session-Based Recommendation – Challenges and Recent Advances. In *KI 2018: Advances in Artificial Intelligence*, Frank Trollmann and Anni-Yasmin Turhan (Eds.). Springer International Publishing, 3–7.

[21] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. *Proceedings of the Eleventh ACM Conference on Recommender Systems* (2017).

[22] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'08).*, 426–434. https://doi.org/10.1145/1401890.1401944

[23] Alexandra Lamont and Rebecca Webb. 2010. Short- and long-term musical preferences: what makes a favourite piece of music? *Psychology of Music* 38, 2 (2010), 222–241. https://doi.org/10.1177/0305735609339471

[24] Cyril Laurier, Mohamed Sordo, Joan Serra, and Perfecto Herrera. 2009. Music Mood Representations from Social Tags.. In *ISMIR*. 381–386.

[25] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-Based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 1419–1428. https://doi.org/10.1145/3132847.3132926

[26] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research* 18, 185 (2018), 1–52. http://jmlr.org/papers/v18/16-558.html

[27] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *CoRR* abs/1803.09587 (2018). arXiv:1803.09587 http://arxiv.org/abs/1803.09587

[28] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Empirical Analysis of Session-Based Recommendation Algorithms. *arXiv preprint arXiv:1910.12781* (2019).

[29] Douglas W. Oard and Jinmook Kim. 1998. Implicit Feedback for Recommender System. https://www.aaai.org/Library/Workshops/1998/ws98-08-021.php

[30] Sung Park, Sangkeun Lee, and Sang-goo Lee. 2011. Session-Based Collaborative Filtering for Predicting the Next Song. (05 2011). https://doi.org/10.1109/CNSI.2011.72

[31] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4, Article Article 66 (July 2018), 36 pages. https://doi.org/10.1145/3190616

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. Association for Computing Machinery, New York, NY, USA, 811–820. https://doi.org/10.1145/1772690.1772773

[33] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive Neural Architecture Incorporating Song Features for Music Recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 417–421. https://doi.org/10.1145/3240323.3240397

[34] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[35] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (Dec. 2005), 1265–1295.

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *CoRR* abs/1409.3215 (2014). arXiv:1409.3215 http://arxiv.org/abs/1409.3215

[38] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-Based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 17–22. https://doi.org/10.1145/2988450.2988452

[39] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H. Chi. 2019. Towards Neural Mixture Recommender for Long Range Dependent User Sequences. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1782–1793. https://doi.org/10.1145/3308558.3313650

[40] Roberto Turrin, Andrea Condorelli, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana. 2015. Large scale music recommendation. In *Workshop on Large-Scale Recommender Systems (LSRS 2015) at ACM RecSys*.

[41] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music Listening and Playlists Dataset. In *RecSys Posters*.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[43] João Vinagre, Alípio Mário Jorge, and João Gama. 2015. An overview on the exploitation of time in collaborative filtering. *WIREs Data Mining and Knowledge Discovery* 5, 5 (2015), 195–215. https://doi.org/10.1002/widm.1160

[44] Sergey Volokhin and Eugene Agichtein. 2018. Understanding Music Listening Intents During Daily Activities with Implications for Contextual Music Recommendation. In *Proceedings of the 2018 Conference on Human Information Interaction Retrieval (CHIIR '18)*. Association for Computing Machinery, New York, NY, USA, 313–316. https://doi.org/10.1145/3176349.3176885

[45] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for Next Basket Recommendation. https://doi.org/10.1145/2766462.2767694

[46] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. *CoRR* abs/1902.04864 (2019). arXiv:1902.04864 http://arxiv.org/abs/1902.04864

[47] Bo Wu, Wen-Huang Cheng, Yongdong Zhang, Qiushi Huang, Jintao Li, and Tao Mei. 2017. Sequential Prediction of Social Media Popularity with Deep Temporal Context Networks. *CoRR* abs/1712.04443 (2017). arXiv:1712.04443 http://arxiv.org/abs/1712.04443

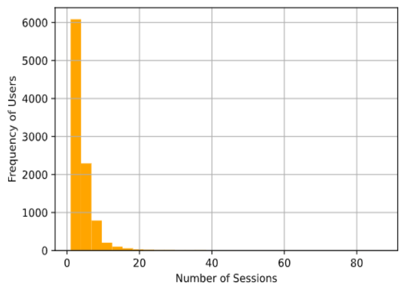
[48] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 729–732. https://doi.org/10.1145/2911451.2914683

[49] Ethan Zhang and Yi Zhang. 2009. *Recall*. Springer US, Boston, MA, 2348–2348. https://doi.org/10.1007/978-0-387-39940-9_479

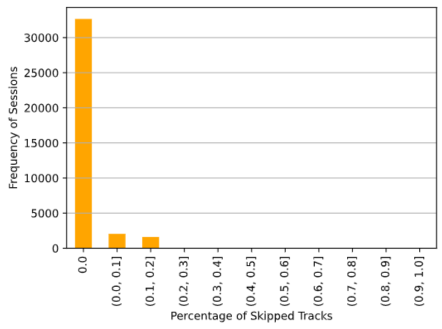
[50] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.07435 (2017). arXiv:1707.07435 http://arxiv.org/abs/1707.07435

A 30MUSIC DATA VISUALISATION

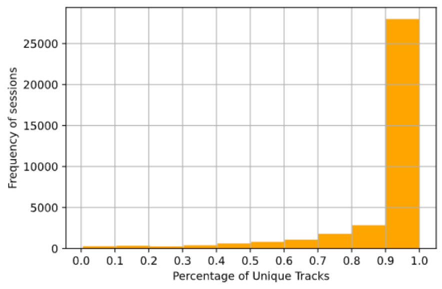
(a) Sessions/user



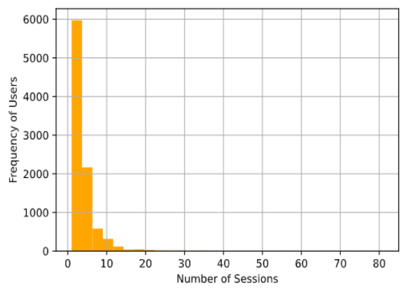
(b) %skipped tracks/session



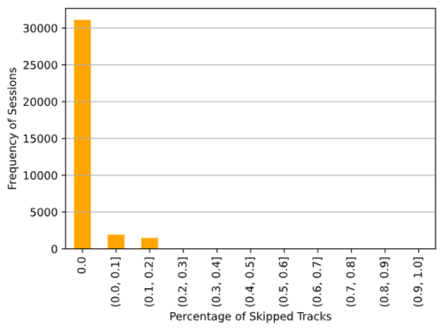
(c) %unique tracks/session



(d) Sessions/user



(e) %skipped tracks/session



(f) %unique tracks/session

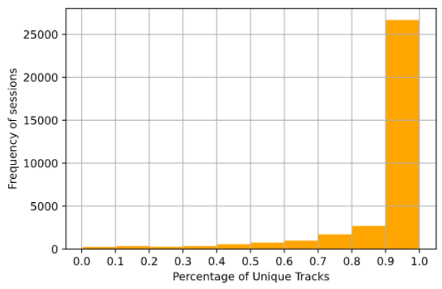


Figure 9: Graphs for the second and third **30Music** dataset slice. Figures 9a and 9e show the distribution histograms of the number of sessions by frequency of users; figures 9b and 9e show the distribution bar charts of the percentage of skipped tracks by frequency of sessions; and figures 9b and 9f show the distribution histograms of percentage of unique tracks per frequency of sessions.

B 30MUSIC BASELINE HYPERPARAMETERS

In addition to the hyperparameters presented in table 3, we employ the hyperparameters in table 7 for the remaining baselines on the 30Music dataset. We use the session-rec framework presented by Ludewig et al. [28] (available at https://github.com/rn5l/session-rec), and thus also adapt their default parameters. Note that we have modified the framework to support our Python 3.5+ work environments as well as corrected configuration, loading, and evaluation issues present in the original code. Our code is available at https://github.com/x775/SW10-Ludewig.

GRU4REC	hidden dropout = 0.1 learning rate = 0.08 momentum = 0.1 loss function = `top1-max` final activation function = `linear`
GRU4REC2	hidden dropout = 0.6 learning rate = 0.08 momentum = 0.0 constrained embedding = `True` loss function = `bpr-max` final activation function = `elu-0.5`
SKNN	neighbours = 100 similarity measure = `cosine`
VSKNN	neighbours = 100 similarity measure = `cosine`

FMPC	learning rate = 0.05
	regularisation = 0.0025
	annealing = 1.0
	initial sigma = 1
	k_cf = 100
	k_mc = 100
	adaptive_sampling = True

FOSSIL	learning rate = 0.05
	regularisation = 0.0025
	annealing = 1.0
	initial sigma = 1
	markov chain order = 1
	alpha = 0.2
	neighbours = 100

Table 7: Hyperparameters used for baselines on the 30Music dataset

C SKNN AND SKNN-SKIPS HYPERPARAMETER TUNING

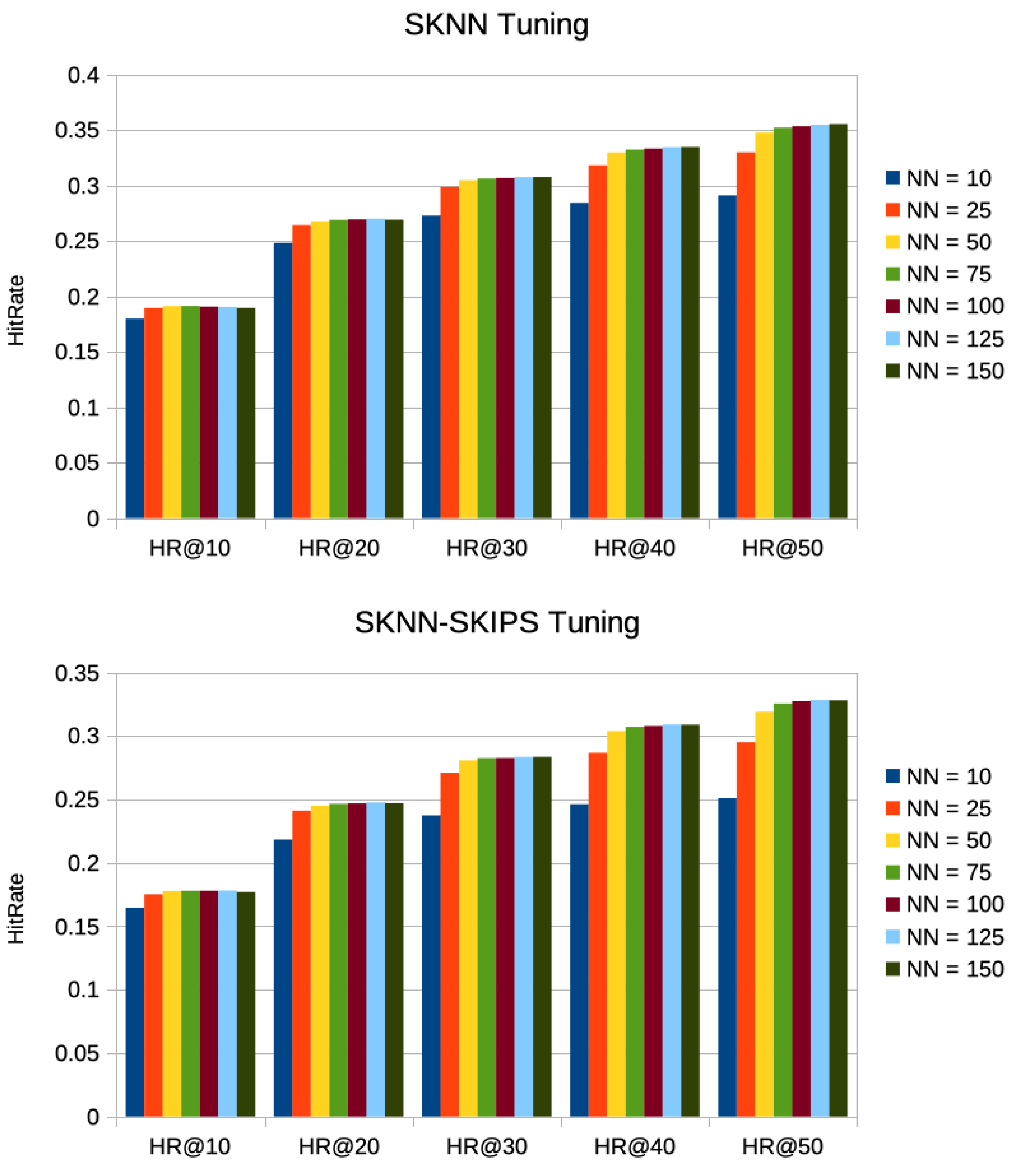


Figure 10: Hit rate performances for SKNN (top) and SKNN-SKIPS (bottom) for 10-150 neighbours on the Lastfm-1K dataset.