Contextual and Sequential User Embeddings for Large-Scale Music Recommendation

Casper Hansen*
University of Copenhagen
c.hansen@di.ku.dk

Christian Hansen*
University of Copenhagen
chrh@di.ku.dk

Lucas Maystre
Spotify
lucasm@spotify.com

Rishabh Mehrotra
Spotify
rishabhm@spotify.com

Brian Brost
Spotify
brianbrost@spotify.com

Federico Tomasi
Spotify
federicot@spotify.com

Mounia Lalmas
Spotify
mounia@acm.org

ABSTRACT

Recommender systems play an important role in providing an engaging experience on online music streaming services. However, the musical domain presents distinctive challenges to recommender systems: tracks are short, listened to multiple times, typically consumed in sessions with other tracks, and relevance is highly context-dependent. In this paper, we argue that modeling users' preferences at the beginning of a session is a practical and effective way to address these challenges. Using a dataset from Spotify, a popular music streaming service, we observe that *a*) consumption from the recent past and *b*) session-level contextual variables (such as the time of the day or the type of device used) are indeed predictive of the tracks a user will stream—much more so than static, average preferences. Driven by these findings, we propose CoSeRNN, a neural network architecture that models users' preferences as a sequence of embeddings, one for each session. CoSeRNN predicts, at the beginning of a session, a preference vector, based on past consumption history and current context. This preference vector can then be used in downstream tasks to generate contextually relevant just-in-time recommendations efficiently, by using approximate nearest-neighbour search algorithms. We evaluate CoSeRNN on session and track ranking tasks, and find that it outperforms the current state of the art by upwards of 10% on different ranking metrics. Dissecting the performance of our approach, we find that sequential and contextual information are both crucial.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Music retrieval**.

KEYWORDS

Music Recommendation, User Embeddings, Context, Sequence

*This work was done while the first two authors were at Spotify.

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412248

ACM Reference Format:
Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412248

1 INTRODUCTION

Recommender systems are essential for providing an engaging experience and for helping users navigating the vast amounts of content available in online services. Successful recommender systems have to accurately model each user's individual preferences, such that the most relevant content can be presented to the user. In this work, we consider online music streaming services, which have become increasingly popular in the past decade. By letting users access millions of tracks at the click of a button, they are contributing to democratizing access to music. However, in contrast to other well-studied domains (such as recommending books, movies or clothes), music recommender systems face distinctive challenges [30]. Tracks are short, and therefore often consumed together with other tracks; we refer to such a set of tracks listened to in short succession as a session. A given session often contains tracks from the user's recent consumption history [1], suggesting that the *sequence* of sessions captures essential information about users' changing preferences. Additionally, the relevance of tracks is highly contextual, and preferences depend, among others, on the time of the day and the current season [24]. We seek to embrace these distinctive characteristics to produce a better, more accurate model of user preferences. We focus on the following problem: for a given user, we are interested in predicting, at the beginning of a session, which tracks the user will listen to during the session. We assume that we have access to the user's *past consumption* and to information about the *current context*. This formulation of the problem enables generating recommendations that are not only matching the user's global tastes, but are also tailored to the specific context and situation they currently find themselves in. While generic context-aware recommender systems have been studied in the past [28], little work has been focused on music recommendation. We aim to address this gap.

We begin our investigation by exploring a dataset from an online music streaming service, containing detailed information about the tracks streamed during a two months period for a sample of 200,000 users. We define context as the time of the day (morning, afternoon,

etc.) and the device used to access the service (mobile, desktop, etc.) We find clear evidence that, for a given user, sessions sharing the same context (e.g., sessions happening in the morning) are more similar to each other than to sessions from a different context. We also find that the more the tracks a user listens to during a session deviate from their average preferences, the more likely they are to hit the *skip* button—a negative satisfaction signal. Deviations from the user's average preferences may be due to contextual changes (such as morning vs. evening), but also to preference drifts that are captured in recent sessions. These observations are consistent with our hypothesis: accurately modeling sequential and context-specific intents is important to ensure high user satisfaction across all sessions.

Taken together, these findings support the idea of learning sequence and context-aware models of user preferences. To this end, we introduce *CoSeRNN*.[1] Our starting point is a vector-space embedding of tracks, where two tracks are close in space if they are likely to be listened to successively. Given this space, CoSeRNN models user preferences as a sequence of context-dependent embeddings (points in the track space), one for each session. At its core, it is a variant of a recurrent neural network that takes as input, for each session, the current session context and a representation of the user's past consumption. Given these, the model is trained to output an embedding that maximizes the cosine similarity to the tracks played during the session. Interestingly, we find that the most effective way to produce this embedding is to fuse a long-term, context-independent vector (intuitively capturing a user's average tastes) with a sequence and context-dependent offset (capturing current and context-specific preferences).

We evaluate our approach experimentally against multiple competing baselines on *a*) a session ranking task, where the goal is to discriminate between the current session and previous ones, and *b*) a track ranking task, where the goal is to predict which tracks a user will listen to in the current session. Our approach performs significantly better than competing approaches: we observe gains upwards of 10% on all ranking metrics we consider. We study these results in depth. First, we break them down by context, and observe that CoSeRNN exhibits the biggest gains on infrequent contexts. Second, we perform an ablation study and discover that combining both sequential and contextual information is crucial to achieve high accuracy. In summary, CoSeRNN sucessfully demonstrates the benefits of modeling preferences at the session level.

Setting predictive performance aside, we believe that our design choices also highlight an interesting point in the recommender systems solution space. Broadly-speaking, our method falls within the realm of representation learning, which postulates that low-dimensional embeddings provide an effective way to model users and items [20, 21]. Whereas most of the work in this area has been focused on *jointly* learning user and item embeddings, we choose a different path, and instead take advantage of an existing track embedding space. By decoupling track and user embeddings, and learning the latter based on the former, we ensure interoperability with other models seeking to address problems that are distinct from contextual or sequential recommendations—our focus in this paper. In addition, and similarly to [21], our model does not seek to *directly* predict the individual tracks inside a session; instead, it generates a session-level user embedding, and relies on the assumption that tracks within the session lie inside a small region of the space. Relevant tracks can then be found efficiently using approximate nearest-neighbor search [2]. This choice enables our method to scale to millions of tracks effortlessly.

Outline & Contributions. After briefly discussing related work (Section 2) and describing our dataset (Section 3), we investigate the following two research questions.

RQ1 *Does music consumption depend on context?* By means of simple analyses, we show clear evidence of contextual patterns in music consumption (Section 4) and thus answer the question in the affirmative.

RQ2 *Can sequential and context-dependent user embeddings better anticipate a user's music consumption?* We address this question by presenting CoSeRNN, a sequence and context-aware model of user preferences (Section 5), and demonstrate that it can achieve state-of-the-art results on several prediction tasks (Section 6). We make our implementation of CoSeRNN publicly available.[2]

2 RELATED WORK

Recommender systems can be broadly categorized as using *explicit* or *implicit* feedback, depending on how users are assumed to indicate their preferences [20]. They can be further categorized based on which information is available besides user feedback. This includes content-based [25], sequence-aware [27], context-aware [28], and collaborative filtering recommender systems [20]. In practice, the ideas underlying these various approaches can be combined to match the exact problem setting at hand. Traditionally, matrix and tensor factorization approaches have been widely successful for recommendation tasks [10, 20], but recently deep learning based techniques [37], specifically recurrent neural networks, have received increasing interest, due to their ability to model the sequential nature of user-item interactions effectively [3, 9, 21, 29, 35]. Our model is based on a recurrent neural network, and as such shares similarities with this line of work.

Of particular relevance to us is *session-aware* recommendation [27], where the focus is on modeling users' preferences and intents during a specific session. Early work on session recommendation utilized Markov chains to predict the next action within a session [40], and was later extended to Markov decision processes [31]. However, if higher order models are used, then the state space grows too large and becomes impractical. To this end, recurrent neural network models have proven useful, especially for sequential click prediction tasks [15, 38], where parallel mini-batches and ranking losses lead to large performance increases over earlier work. Other approaches focus on directly exploiting user behaviour to improve performance, e.g., by explicitly modelling repeat consumption [1, 7, 29]. In contrast to previous work, we consider a slightly different setting: we seek to model user preferences at the beginning of a session but *before* observing any user interaction. We also assume access to *explicit* information about the context.

[1] **Co**ntextual and **Se**quential **R**ecurrent **N**eural **N**etwork

[2] Code for CoSeRNN available at https://github.com/spotify-research/cosernn.

The session-based recommender systems described above are similar to the generic next-item recommendation setting [9, 19, 36], but typically session-based methods directly exploit the similarities between items within the same session. Related tasks include predicting the first item in the next session [29], as well as predicting all items in the next session (also known as next-basket recommendation in the e-commerce domain) [34]. However, as the number of possible items grows, predicting every individual item in a session becomes intractable due to the combinatorial number of possibilities. In this paper, we overcome this problem by representing sessions compactly using embeddings. Thus, our setting is similar to the next-item recommendation setting, since ultimately we predict a single embedding representing an entire session.

The idea of predicting the embedding of the next item, rather than the item itself, has been recently investigated [21]. Given an embedding, recommendations are then based on inexpensive similarity computations, which allows for very large item pools [2]. JODIE [21] learns dynamic user and item embeddings through a coupled recurrent neural network, where the item embeddings are based on learning a future user embedding projection. In addition to dynamic embeddings, JODIE also uses static embeddings that represent the long-term stationary properties of users and items, respectively. In contrast to JODIE, our approach does not explicitly project the user embedding, but rather learns it implicitly in the recurrent neural network component of our model. Also, we represent long-term user properties as an explicit combination of all previous sessions, and let the model learn a sequence and context-dependent *offset vector* that is fused with the long-term representation.

2.1 Music Recommendation

Music recommender systems present different challenges compared to recommender systems applied to movies, books, and other products [30]. The major differences are with regards to the duration of an item (e.g., a song is typically much shorter than a movie) and consumption type and intent (music streaming is inherently sequential and highly contextual). Some of these unique characteristics have previously been explored in the setting of playlist generation [4, 8]. Others have investigated the effects of context [6, 14, 33], location [17], and even the weather [26]. However, these studies usually rely on small-scale datasets and do not explore the impact of context on recommendation accuracy and performance. In contrast, our work takes advantage of a large-scale dataset from a leading music streaming service, and evaluates the predictive performance of a context and sequence-aware model on concrete recommendation tasks. Finally, we note that whereas we focus on the sequence of sessions in this work, prior work on the publicly available Spotify Music Streaming Sesssions Dataset [5] addressed the problem of within-session sequencing.

3 DATASET

In this section, we introduce a dataset from Spotify, an online music streaming service. Through Spotify, users have on-demand access to millions of music tracks.[3] We focus on so-called *premium* users, who enjoy an unrestricted, ad-free streaming experience. We consider the listening history of a sample 200,000 users from April 1st to May 31st 2019. We group listening history into sessions, where we define a session as the set of music tracks consumed in a given time interval, such that two sessions are separated by at least 20 minutes of inactivity. On average, users in the dataset have 220 sessions during the two-month period, and each session consists of 10 tracks on average.

[3] See: https://newsroom.spotify.com/company-info/.

Table 1: Summary of the features extracted from a session t contained in the dataset.

Symbol	Description	Domain
D_t	Day of the week	$\{1, \ldots, 7\}$
H_t	Time of the day	$\{0, \ldots, 23\}$
Y_t	Device	$\mathcal{Y}$
N_t	Number of tracks in session	$\mathbb{N}_{>0}$
Δ_t	Time since last session	$\mathbb{R}_{>0}$
z_t	Stream source	$\mathcal{Z}$
$\boldsymbol{s}_t$	Session embedding, all tracks	$\mathbb{R}^{40}$
$\boldsymbol{s}_t^+$	Session embedding, *played* only	$\mathbb{R}^{40}$
$\boldsymbol{s}_t^-$	Session embedding, *skipped* only	$\mathbb{R}^{40}$

3.1 Session-Level Information

Each session is annotated with detailed information, including *a*) the set of tracks played during the session, *b*) which tracks were skipped, *c*) the stream source (user playlist, top charts, etc., collectively denoted $\mathcal{Z}$), *d*) a timestamp representing the start of the session, and *e*) the device used to access the service. We process this information into a set of features, presented in Table 1. The contextual features available at the beginning of the session, D_t, H_t and Y_t, can be categorized into two types:

- **Time context.** We use day of the week D_t and time of the day H_t. Note that even though, in Section 4, we partition sessions by using D_t only, *both* features are used for the model described in Section 5.
- **Device context.** In addition, we consider the device Y_t used by the user to access the service at the beginning of a session. We restrict ourselves to the major devices: $\mathcal{Y}$ = {mobile, desktop, speaker, web, tablet}.

We choose these features as our contextual variables because we believe that they are both important and widely available. Nevertheless, our framework is independent of the particular choice of context and other information (either explicit or implicit) such as mood, activity, or intent can be integrated effortlessly, if available.

The music listened by the user during the session is summarized using three 40-dimensional session embeddings, $\boldsymbol{s}_t$, $\boldsymbol{s}_t^+$, $\boldsymbol{s}_t^-$. These are described in Section 3.3, building upon the description of track embeddings.

3.2 Track Embedding

We embed tracks in a latent semantic space using the *word2vec* continuous bag-of-word model [23] on a set of user-generated playlists. In short, the model learns 40-dimensional real-valued unit-norm embedding for each track, such that two tracks that are likely to co-occur in a playlist are close to each other in the embedding

space, and vice-versa. The similarity between two tracks can then be computed simply by using the cosine similarity between their embeddings. The specific embedding space that we use has previously been shown to work well for music recommendation [22].

It should be noted that, in principle, track embeddings and user embeddings could be learned jointly. By decoupling the two, we simplify the development of multiple models with different goals and improve the scalability of our approach, as discussed in Section 1.

3.3 Session Embedding

The way we represent sessions builds on the track embedding model. In fact, we represent a session simply as an average of the embedding of the tracks it contains. The assumption is that, within a session, tracks cluster around a small region of the embedding space,[4] and as such the average track embedding provides a compact summary of the session's content. We consider three embedding variants for a given session t, all normalized to unit length: $\boldsymbol{s}_t$ represents the average of *all* tracks, while $\boldsymbol{s}_t^+$ and $\boldsymbol{s}_t^-$ represent the average of *played* and *skipped* tracks, respectively. Considering played and skipped tracks separately provides a more detailed picture of session-level user preferences.

Our definition of session embedding is computationally-efficient: if we also model user preferences by using a unit-norm vector in the same embedding space (as we do in Section 5), we can compute the cosine similarity to a given session's embedding using a dot product. Due to the distributive property of the dot product, the result can be thought of as the average relevance of each track to the user—all using a single dot product.

Finally, we note that, for long sessions, it is no longer clear whether the context stays constant throughout its duration, and whether the tracks played at the end of the session are related to the ones at the beginning. For this reason, we deliberately consider only the first 10 tracks within a session (equal to the average session length), and discard the rest. We are thus effectively understanding, modeling and predicting the *beginning* of a session.

4 EXPLORATORY ANALYSES

In this section, we demontrate the need for contextual models by studying the influence of context on music consumption. We aim to answer the following sub-questions to RQ1, related to context in music consumption:

- 4.1 How are sessions distributed according to context, and what is the proportion of users experiencing each type of context?
- 4.2 Does music consumption vary depending on context?
- 4.3 How similar are sessions across and within different types of contexts?

These questions provide empirical evidence for considering context in music recommendation. Finally, we investigate how user satisfaction relates to how different a session is from a user's average preferences (see Section 4.4)

4.1 Context Distribution Across Sessions

The majority of sessions, as shown in Figure 1, are happening in the afternoon (12pm-5pm) and evening (5pm-8pm). The remaining (46.4%) of sessions are spread out over the remaining time contexts of early morning (6am-9am), morning (9am-12am), late evening (9pm-1am), and night (1am-6am). It is interesting to consider that most users have sessions spanning all types of time contexts, except for the *night* context, which relates to only 76.5% of users. For the device context, we observe that 88.3% of sessions are happening on mobile devices, 8.6% on desktop, and the remaining ones are split across speaker, web, and tablet. However, similar to the time context, a significant amount of users do have at least one session in one or more of the non-mobile devices. These findings highlight that users consume music across multiple contexts, and that even minority contexts are important to consider, since a large number of users do experience those at some point.

[4]We validate this assumption empirically using track and session ranking tasks in Section 6.

4.2 Music Consumption and Context

We investigate if diversity in music consumption depends on context. We collect all tracks appearing in each specific context, and compute the pairwise cosine similarities between tracks within each context. Note that even minority contexts (such as web and tablet) contain millions of plays, meaning that the distributions are well captured for all contexts. Figure 2a illustrates the distributions of pairwise similarities by using boxplots. For the time context, we see that minority contexts, such as early morning and night, have significantly larger variability compared to majority sessions such as afternoon and evening. We observe a similar trend for the device context for all non-mobile contexts compared to mobile. This suggests that users have largely different needs in minority contexts, and as such the user embedding needs to incorporate context in order to reliably estimate user needs.

4.3 Session Similarity and Context

In the previous section, we performed a global analysis across all tracks from every user within a specific context. Now, we analyse if, for a given user, their sessions within the same context are more similar across different contexts. For each user and each session (the *source*), we find the nearest session (the *target*) among all the user's other sessions, and store both the source's and the target's contexts. We then aggregate all the pairs and compute the empirical distribution of the target's context type, conditioned on the source's context type. To correct for the bias induced by the users' distinctive usage patterns (some users might use the service more in some contexts than in others), we subtract from this distribution the marginal probability of each pair of contexts (or, equivalently the empirical distribution obtained when sampling *source* and *target* uniformly at random among the same user's sessions).

Figure 3 displays the result in the form of heatmaps. The positive diagonal tells us that sessions sharing the same context are indeed more similar than sessions sampled at random. Additionally, for time contexts, those occurring close to each other (e.g., night and evening) also often have small positive values, indicating that sessions even from consequent contexts are more similar than random sessions. This analysis highlights that sessions with the same context do share some similarities, which can be exploited to learn better performing contextual user embeddings.

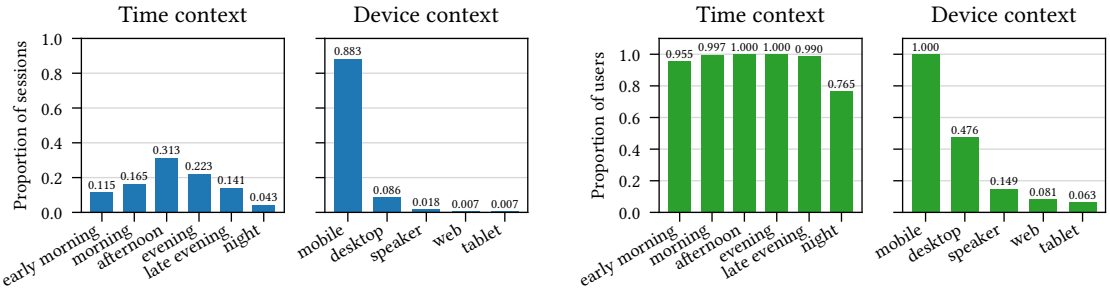
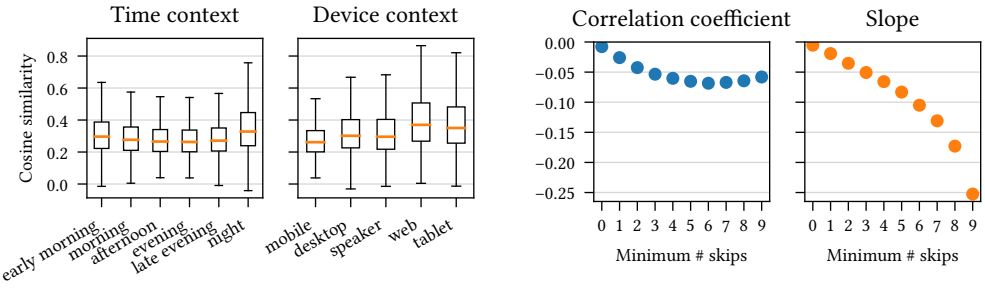


Figure 1: Left: histograms of session context. Right: proportion of users with at least one session within a context.



(a) Homogeneity within contexts.

(b) Cosine similarity and skip rate.

Figure 2: Left: boxplots showing the distribution of pairwise cosine similarity for all tracks occurring within each context. Right: Pearson correlation and regression slope of the relation between skip rate and user-session cosine similarity.

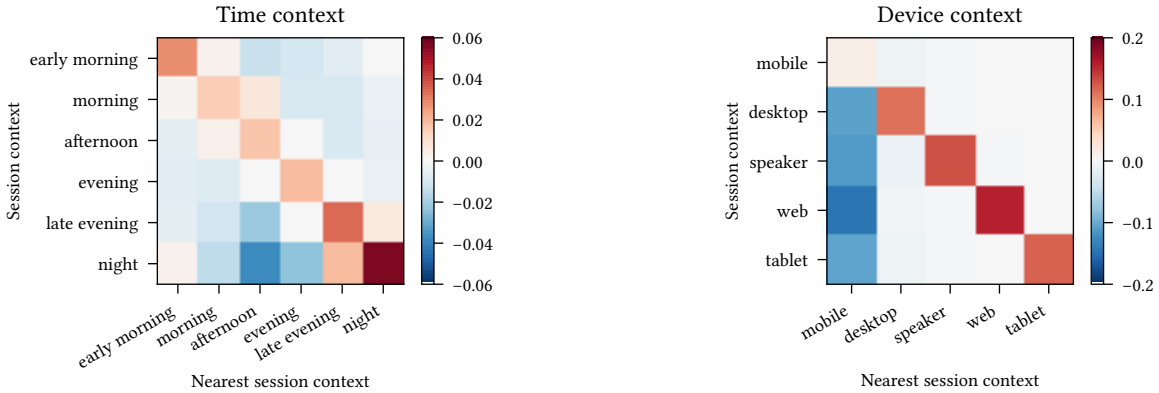


Figure 3: Visualization of the probability difference between sampling sessions ranked by their pairwise cosine similarity or randomly. A positive number corresponds to that pair of context types being more similar than a randomly sampled session.

4.4 Contextual Preferences and Skip Rate

In the previous analyses we found evidence of context being useful for providing a more accurate picture of users' preferences. Now, we consider the influence of a better match between user and session embeddings (i.e., higher cosine similarity) on user satisfaction. As a proxy for satisfaction, we measure the skip rate, i.e., the percentage of skipped tracks within a session. For the purposes of this analysis, we define the user embedding as an average of the embeddings of all their previous sessions. For each session we record the cosine similarity between the user embedding and the current session's embedding, as well as the skip rate of the session.

Figure 2b shows the Pearson correlation coefficient and regression slope between the skip rate and user-session cosine similarity, as a function of the minimum skip rate. By considering sessions with at least k skips (x-axis in the figure), we filter out sessions with low activity, since a very low amount of skips may simply be due to the user not being actively engaged. We observe that both the correlation coefficient and regression slope are negative. This means that, as users skip more often, the user-session similarity decreases. Additionally, both the correlation coefficient and regression slope generally decrease the larger the minimum number of skips is. We expect that, if we were able to anticipate these "unusual" sessions (i.e., sessions that deviate significantly from a users' average

preference), we might be able to improve user satisfaction. As we will demonstrate in the next sections, sequence and context-aware models enable us to achieve that goal.

5 CONTEXTUAL AND SEQUENTIAL MODEL

The previous section established the importance of context for understanding users' behaviors. Building on these findings, we now present CoSeRNN, a user-embedding model that captures contextual and sequential preferences at the session level. The aim is to predict, at the very beginning of a session (without observing any explicit action from the user), which tracks will be played during the session, based on features derived from the past consumption history and the current context. Section 5.1 presents the architecture of our model and Section 5.2 discusses the procedure we use to train it.

5.1 Model Architecture

For conciseness, we consider a single user. For a given a session index t, we denote the predicted session-level user embedding as $\boldsymbol{u}_t \in \mathbb{R}^{40}$, and the observed (ground-truth) session embeddings as $\boldsymbol{s}_t^-, \boldsymbol{s}_t^+ \in \mathbb{R}^{40}$, as defined in Section 3.3. The model is trained to maximize the similarity between $\boldsymbol{u}_t$ and $\boldsymbol{s}_t^+$ (this will be made precise in Section 5.2). A diagram of the architecture of our proposed model, CoSeRNN, is provided in Figure 4. At a high level, CoSeRNN models $\boldsymbol{u}_t$ as follows. It uses features about the current context (such as time of the day and device) and features about the last session as input to two RNNs, representing *play* and *skip* behavior. These RNNs combine the input with a latent state, capturing sequential dependencies in the user's consumption habits. Finally, the outputs of the two RNNs are combined and fused with a long-term user embedding.

5.1.1 Notation. We denote a dense (fully-connected) neural network layer by $\mathrm{FC}_g(\boldsymbol{x}) = g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ are a weight matrix and a bias vector of suitable dimensions, respectively, and g is an activation function. We consider three such functions, *a*) the identity function $\mathrm{id}(\boldsymbol{x}) = \boldsymbol{x}$, *b*) the elementwise rectifier $\mathrm{ReLU}(\boldsymbol{x}) = [\max\{0, x_i\}]$, and *c*) the softmax function $\mathrm{softmax}(\boldsymbol{x}) = [\exp x_i / \sum_j \exp x_j]$. We denote by $f \circ g(x)$ the composition of f and g evaluated at x, i.e., $f(g(x))$, and by $\boldsymbol{x} \oplus \boldsymbol{y}$ the concatenation of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

5.1.2 Input Layers. We start with two feature vectors,

$$\boldsymbol{f}_t^+ = \boldsymbol{c}_t \oplus \boldsymbol{s}_{t-1}^+ \oplus \begin{bmatrix} N_{t-1} & z_{t-1} & \Delta_t \end{bmatrix}$$
$$\boldsymbol{f}_t^- = \boldsymbol{c}_t \oplus \boldsymbol{s}_{t-1}^- \oplus \begin{bmatrix} N_{t-1} & z_{t-1} & \Delta_t \end{bmatrix}$$

where $\boldsymbol{c}_t$ is a concatenation of one-hot encodings of the contextual variables D_t, H_t and Y_t, and all other symbols refer to Table 1. These are input to the *play* and *skip* pathways of the network, respectively. Prior to passing the features to the RNN, we apply a learned nonlinear transformation. This enables the RNN to better focus on modeling latent sequential dynamics. In particular, we apply the following transformation: $\hat{\boldsymbol{f}}_t^+ = \mathrm{FC}_{\mathrm{ReLU}} \circ \mathrm{FC}_{\mathrm{ReLU}}(\boldsymbol{f}_t^+)$, which corresponds to two fully connected layers with ReLU activations. We obtain $\hat{\boldsymbol{f}}_t^-$ by applying the same transformation to $\boldsymbol{f}_t^-$.

5.1.3 Recurrent Layers. Next, we seek to capture and reuse relevant information from the user's history—beyond the last session. We do so by using an RNN with Long Short Term Memory (LSTM) cells [16], and let $(\boldsymbol{o}_t^+, \boldsymbol{h}_t^+) = \mathrm{LSTM}(\hat{\boldsymbol{f}}_t^+ \mid \boldsymbol{o}_{t-1}^+, \boldsymbol{h}_{t-1})$, where $\boldsymbol{o}_t^+$ is the output and $\boldsymbol{h}_t^+$ the hidden state.[5] Similarly, we obtain $(\boldsymbol{o}_t^-, \boldsymbol{h}_t^-)$ from $\hat{\boldsymbol{f}}_t^-$. We learn to combine the outputs $\boldsymbol{o}_t^+$ and $\boldsymbol{o}_t^-$, and then obtain the sequence and context-dependent part of the user embedding, $\hat{\boldsymbol{u}}_t$, as

$$\boldsymbol{o}_t = \mathrm{FC}_{\mathrm{ReLU}} \circ \mathrm{FC}_{\mathrm{ReLU}}(\boldsymbol{o}_t^- \oplus \boldsymbol{o}_t^+), \qquad \hat{\boldsymbol{u}}_t = \mathrm{FC}_{\mathrm{id}}(\boldsymbol{o}_t).$$

5.1.4 Fusion with Long-term User Embedding. A long-term, context-independent embedding is able to explain the general preferences of a user relatively well [22]. We build upon this observation, and enable our RNNs to focus on learning session-specific *deviations* from a long-term user embedding $\bar{\boldsymbol{u}}_t$, defined as a weighted average of all previous session embeddings,

$$\bar{\boldsymbol{u}}_t \propto \sum_{t'=1}^{t-1} \frac{t'}{t-1} \boldsymbol{s}_{t'}, \tag{1}$$

normalized such that $\|\bar{\boldsymbol{u}}_t\| = 1$. To fuse $\bar{\boldsymbol{u}}_t$ and $\hat{\boldsymbol{u}}_t$, we learn attention weights based on the RNN output, such that uncertain RNN estimates can default to the long-term embedding. We compute the attention weights and use those to produce the final user embedding as $\boldsymbol{u}_t = \hat{\beta}_t \hat{\boldsymbol{u}}_t + \bar{\beta}_t \bar{\boldsymbol{u}}_t$, where $\begin{bmatrix} \hat{\beta}_t & \bar{\beta}_t \end{bmatrix} = \mathrm{FC}_{\mathrm{softmax}}(\boldsymbol{o}_t)$.

5.2 Training and Hyperparameter Tuning

To learn the parameters of the model, tune hyperparameters and evaluate the performance of our model, we split the dataset of Section 3 into training, validation and test sets, respectively. The test set consists of all the sessions of the last two weeks of the dataset, and the validation set of the 5 days prior to the beginning of the test set.

5.2.1 Loss function & Optimization. Our model is trained to maximize the cosine similarity between the predicted embedding $\boldsymbol{u}_t$ and the observed one $\boldsymbol{s}_t$. Because both embeddings are unit-norm, the cosine similarity can be computed simply by using a dot product. Formally, letting $\mathcal{D}$ be a training set consisting of pairs (i, t), where i denotes the user and t the session, the loss function is defined as $\ell = \sum_{(i,t)\in\mathcal{D}} \left(1 - \boldsymbol{u}_{it}^\top \boldsymbol{s}_{it}^+\right)$, where $\boldsymbol{u}_{it}$ and $\boldsymbol{s}_{it}^+$ refer to the t-th session of user i. We minimize ℓ using stochastic gradient descent with mini-batches, and find that the Adam optimizer [18] works well, converging in a few tens of epochs.

5.2.2 Hyperparameter Tuning. We use the validation set to tune the learning rate $\lambda \in \{0.001, 0.0005, 0.0001\}$, LSTM cell sizes $d \in \{100, 200, 400\}$, and batch sizes $m \in \{128, 256, 512\}$, and keep fully connected layers fixed to size 200. Optimal performance is achieved by setting $\lambda = 0.0005, d = 400, m = 256$.

6 EXPERIMENTAL EVALUATION

We evaluate the predictive performance of our model on the music sessions dataset described in Section 3. We first present competing approaches (Section 6.1), then we describe two ranking tasks that we use to evaluate our model (Section 6.2) and provide detailed results (Section 6.3). Finally, we perform an ablation study and shed

[5] At $t = 0$, the hidden state is initialized using a learned embedding that depends on the user's age.

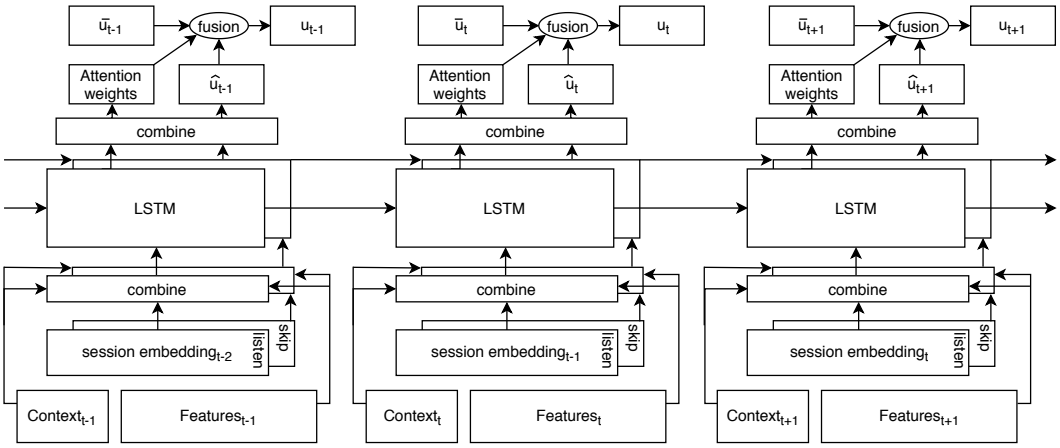


Figure 4: Overview of the CoSeRNN model. The model captures users' sequence-dependent and context-dependent preferences using information available at the beginning of a session.

light on how the different components of our model contribute to its predictive performance (Section 6.4).

6.1 Baselines

We evaluate the performance of several baselines and ground-truth approaches. Our aim is to *a*) understand the various metrics we consider in terms of lower-bounds (achieved by trivial models) and upper bounds (ground-truth), and *b*) tease apart the impact of modelling contextual and sequential effects. We consider the following six baselines.

Last, any cxt This simple baseline predicts the current session embedding using the vector of the last session (irrespective of that session's context), that is, $\boldsymbol{s}^{+}_{t-1}$. The underlying assumption is that the session vector does not depend on context but that it may change quickly over time.

Last, same cxt Similar to the previous one, except that it uses the vector of the last session whose context is identical to the current one.

Avg, any cxt The current session is modeled as a weighted average of all past session vectors (irrespective of their contexts), similarly to the long-term user embedding in Equation (1).

Avg, same cxt Consists of a weighted average of past sessions as for the previous baseline, except that only sessions with a context that is identical to the current one are considered.

Popularity The predicted current session vector is equal to that of the past session containing the most popular tracks.

JODIE The state-of-the-art embedding prediction model of Kumar et al. [21]. JODIE takes both contextual and sequential effects into account. To match our setup, we leave the track embedding fixed (i.e., we use the existing pretrained embeddings), and use multiple RNNs to represent combinations of all, skipped, and listened parts of the session (similarly to our model). We also use the cosine distance as loss function, as we found that using the ℓ_2-loss (as presented in their paper) obtained inferior results in our setting.

RRN The model of Wu et al. [35]. RRN learns to predict future behavioral trajectories using contextual and sequential information. We tune the parameters used in the original paper and adapt it to our setting by changing the objective function to be the same as that of CoSeRNN (see Section 5.2.1). We replace the original rating prediction layer to have an output of 40 (our embedding size) rather than 1 (their rating score). Similar to JODIE, we leave the embedding of tracks fixed and use multiple RNNs for representing all, skipped, and listened parts of the session.

LatentCross The model of Beutel et al. [3]. LatentCross introduces a method for modulating the state of an RNN model with contextual features. We adapt it to our setting by changing the objective function to be the same as that of CoSeRNN. We also replace the final softmax layer originally used in LatentCross with a feed forward layer of dimension 40 (our embedding size). With this change, LatentCross can be used to generate embedding predictions rather than individual item predictions. Similar to JODIE and RRN, we leave the embedding of tracks fixed, and allow LatentCross to use multiple RNNs for representing all, skipped, and listened parts of the session.

Architectural network choices aside, an important difference between our CoseRNN and the state-of-the-art baselines is in how static (or long-term) user embeddings are combined with recurrent neural network outputs. JODIE uses a static user embedding, RRN learns a stationary user embedding per time step, and LatentCross does not have any. In contrast, CoSeRNN computes a weighted long-term user embedding grounded in a user's actual past consumption, such that the recurrent neural network can focus on learning a sequence and context-dependent *offset* vector, which is fused with the long-term user embedding using attention weights (see Section 5 for further details).

In addition to these baselines, we also examine two oracle variants, to obtain a sense of the difficulty of the various predictive tasks.

Oracle Full The predicted session vector is exactly equal to the (ground-truth) observed one.

Oracle Half The session vector is modeled as the average of half of the tracks of the current session, selected uniformly at random among all the tracks in the session. By using only half of the tracks, it can be seen as a noisy version of Oracle Full, and highlights the inherent variability inside a session.

6.2 Tasks & Metrics

As discussed in Section 5.2, our model is trained to maximize the cosine similarity between the predicted user embedding and observed session embedding. While this loss is attractive from a computational standpoint (being differentiable and smooth), it is merely a useful proxy to the real problem: producing better, more relevant just-in-time recommendations. To assess whether our optimization metric is indeed helping us solving that problem, we evaluate all approaches on two additional tasks.

6.2.1 Session Ranking. The aim here is to measure how well a given approach can discriminate the current session from previous ones. For a given session t, we consider the K session vectors $\{\boldsymbol{s}^+_{t-K+1}, \ldots, \boldsymbol{s}^+_t\}$. We rank these session vectors by decreasing cosine similarity with the predicted user embedding $\boldsymbol{u}_t$, and measure the rank of $\boldsymbol{s}_t$. For $K \in \{20, 50\}$, we report the mean reciprocal rank (MRR) and the average rank.

6.2.2 Track Ranking. Here the aim is to measure how well a given approach can predict the tracks that a user listens to in a given session. Similarly to session ranking, this measures if the approaches are able to adapt to the user's current preferences, but where the individual items are considered in contrast to an aggregated session representation. Given a session t, we consider the set of K distinct tracks a user has listened to most recently (across all previous sessions). For $K \in \{25, 100\}$, we rank these tracks by decreasing cosine similarity with the predicted vector $\hat{\boldsymbol{s}}_t$ and report the mean average precision (mAP) and the average recall@10. If we are able to rank tracks that are contained in the session highly, it means that we can anticipate the user behavior well, e.g., by showing the relevant tracks more prominently (or even start playing them directly).

Experimental Setup. We use the dataset described in Section 3 and the training procedure of Section 5.2. For all methods, when predicting the user embedding of the t-th session, we use all the data up to (but not including) session t.

6.3 Results

Table 2 presents the performance on the test set for the cosine-similarity loss as well as for the various metrics used for the session and track ranking tasks. *CoSeRNN* consistently outperforms the baselines on all metrics across all tasks (all differences are statistically significant using a pairwise two-tailed t-test at the 0.001 level). It is interesting to note that although there is a clear positive correlation between cosine similarity and the other metrics, a higher cosine similarity does not automatically imply better performance on the ranking tasks.

The four models that are optimized on the cosine similarity (*RRN*, *LatentCross*, *JODIE*, and *CoSeRNN*) are clearly superior to the simple heuristic baselines on both ranking tasks. Among all simple baselines, it is worth noting that *Last, any cxt* generally performs the best (except on cosine and on session ranking for $K = 20$), an indication that recency plays an important role in music recommendation; we investigate this further in Section 6.4. This also explains part of the performance gap from the simple baselines to JODIE and CoSeRNN, as they are able to better capture the recency aspect through recurrent neural networks.

The results up to now are averages over all users and test sessions. We now seek to answer the question: does the performance vary across contexts? We plot the relative improvement of *CoSeRNN* over *JODIE* (the state-of-the-art approach) in Figure 5. For conciseness, we only consider a subset of the metrics, and use $K = 50$ and $K = 100$ for the session and track ranking tasks, respectively. Generally, the improvements are consistent across all contexts. Interestingly, some of the contexts that occur infrequently see a comparatively larger relative improvement, such as for the *web* or—to a smaller extent—*night* contexts.

6.4 Ablation Study

We focus on the empirical performance of *CoSeRNN* and seek to understand how different choices affects the model, which we ablate in two different ways: 1) by varying the features given as input to the model, and 2) by processing played and skipped tracks in different ways.

6.4.1 Input Features. We divide features given as input to our model into five groups: the embeddings of the last session $\boldsymbol{s}^+_t$ and $\boldsymbol{s}^-_t$, the current context $\boldsymbol{c}_t$, the number of tracks in the last session N_{t-1}, the time (in seconds) elapsed since the last session Δ_t, and the stream source of the last session z_{t-1}. Additionally, we consider a hypothetical scenario[6] where we have access to the stream source of the *current* session, z_t.

The performance of the corresponding models on the cosine similarity and track ranking tasks is given in Table 3. The current session context is associated with the biggest increase in cosine similarity, which is to be expected as context is highly indicative of the content in a session (as seen in Section 4). In addition, information about the previous session significantly helps improving performance, particularly on the track ranking task. If, in addition to knowing a user device and the time of the day, we know which stream source they intend to stream from, our model predictive performance increases substantially.

6.4.2 Listened vs. Skipped Tracks. The final architecture of our model partitions tracks in a session into two subsets, *played* tracks and *skipped* tracks. To better understand the impact of this particular choice, we compare our model to three alternative variants. The first one does not distinguish between played and skipped tracks, and instead considers the average embedding of *all* tracks in the session. The second one disregards skipped tracks completely and focuses only on played tracks. The last one considers played and skipped tracks separately, but also adds in another RNN pathway that considers the average of all tracks. Table 4 displays the performance attained by each model.

[6]This scenario assumes that we are given partial information about the user intent in the current session. This assumption is realistic in practice, but the trade-off is that we cannot present recommendations immediately at app launch time.

Table 2: Empirical performance of various session-prediction approaches on a music dataset. Best result is highlighted in bold (all differences are statistically significant at the 0.001 level using a paired two-tailed t-test).

		Session ranking				Track ranking			
		$K = 20$		$K = 50$		$K = 25$		$K = 100$	
Model	Cosine	MRR	Rank	MRR	Rank	mAP	Rec@10	mAP	Rec@10
Last, any cxt	0.6527	0.1721	10.0767	0.1133	21.9245	0.3585	0.4394	0.1300	0.1372
Last, same cxt	0.5990	0.2094	9.3128	0.1009	23.5288	0.3398	0.4223	0.1154	0.1156
Avg, any cxt	0.6797	0.1835	10.0882	0.1034	23.7337	0.3485	0.4250	0.1225	0.1291
Avg, same cxt	0.6609	0.2087	9.3365	0.1031	23.4767	0.3476	0.4313	0.1199	0.1239
Popularity	0.4278	0.2012	9.5670	0.0967	24.0233	0.3457	0.4338	0.1165	0.1190
RRN [35]	0.6918	0.1970	9.6689	0.1286	21.2980	0.3794	0.4678	0.1425	0.1594
LatentCross [3]	0.6921	0.1967	9.6669	0.1286	21.2610	0.3794	0.4678	0.1422	0.1592
JODIE [21]	0.6970	0.2079	9.3438	0.1303	21.2258	0.3836	0.4734	0.1450	0.1638
CoSeRNN (ours)	**0.7115**	**0.2319**	**8.6642**	**0.1507**	**19.5288**	**0.4011**	**0.4981**	**0.1574**	**0.1816**
Oracle half	0.7077	0.4723	5.2294	0.3711	11.3131	0.7732	0.8052	0.5824	0.6163
Oracle full	1.0000	0.9985	1.0067	0.9988	1.0068	0.8323	0.8861	0.6220	0.6951

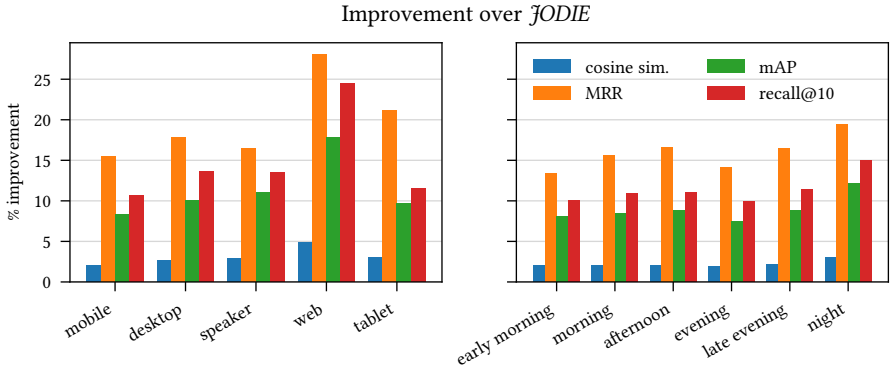


Figure 5: Improvement of *CoSeRNN* over *JODIE*. We compute MRR for $K = 50$ and mAP and recall@10 for $K = 100$.

Table 3: Performance of *CoSeRNN* model variants with access to increasing subsets of input features.

		Track ranking ($K = 100$)	
Features	Cosine	mAP	Recall@10
Last sess. embeddings	0.7062	0.1518	0.1731
+ curr. context	0.7091	0.1527	0.1748
+ # tracks in last	0.7103	0.1569	0.1807
+ time since last	0.7109	0.1569	0.1808
+ last stream source	0.7115	0.1574	0.1816
+ curr. stream source	**0.7313**	**0.1777**	**0.2100**

Table 4: Performance of four *CoSeRNN* model variants that encode the session in different ways.

		Track ranking ($K = 100$)	
Session encoding	Cosine	mAP	Recall@10
All	0.6966	0.1442	0.1623
Plays	0.7103	0.1567	0.1801
Plays + skips	**0.7115**	**0.1574**	**0.1816**
Plays + skips + all	0.7114	0.1569	0.1809

Separating played tracks from skipped tracks is clearly beneficial to model performance. Interestingly, considering skipped tracks in addition to played tracks does bring some performance benefits. Even though the performance benefit is small, this does highlight the dissimilarities between a user skipped and listened tracks, and it helps to improve the model capabilities of understand a user music preferences. Finally, considering the union of played and skipped tracks in addition to the two partitions is unnecessary and does not improve performance.

7 CONCLUSION

In this work, we consider the task of learning contextual and sequential user embeddings suited for music recommendation at the

beginning of a session. To this end, we first perform multiple exploratory analyses, gaining a better understanding of how sessions are distributed according to context, how music consumption varies depending on context, and how context correlates with the tracks within a session. We find that most users experience a diversity of contexts (even though some occur more frequently than others), that sessions belonging to rarely occurring contexts vary the most (in terms of contents), and that sessions with the same context have more similar content.

Driven by these findings, we present CoSeRNN, a recurrent neural network embedding model that learns the sequential listening behaviour of users, and adapts it to the current context. CoSeRNN does this through the combination of *a*) a global long-term embedding that captures a user's long-term music preferences, and *b*) a sequence and context-dependent offset. In contrast to prior methods that require expensive model evaluations to produce recommendations, the approach taken by CoSeRNN enables efficiently generating recommendations by using fast approximate nearest neighbour searches. When evaluated empirically on a large-scale dataset of sessions, CoSeRNN outperforms baseline and state-of-the-art embedding-based approaches by upwards of 10% in session and track recommendation tasks. In future work, hashing-based embedding approaches would be interesting to investigate in our setting, as existing work on content-aware recommendation [12, 39] and similarity search [11, 13, 32] have shown hashing-based approaches to allow large efficiency gains at the cost of a marginal effectiveness reduction.

REFERENCES

[1] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii. 2014. The dynamics of repeat consumption. In *international conference on World wide web*. ACM, 419–430.

[2] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Conference on Recommender systems*. ACM, 257–264.

[3] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *International Conference on Web Search and Data Mining*. 46–54.

[4] G. Bonnin and D. Jannach. 2014. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)* 47, 2 (2014), 1–35.

[5] B. Brost, R. Mehrotra, and T. Jehan. 2019. The Music Streaming Sessions Dataset. In *The World Wide Web Conference*. ACM, 2594–2600.

[6] T. Cebrián, M. Planagumà, P. Villegas, and X. Amatriain. 2010. Music recommendations with temporal context awareness. In *Conference on Recommender systems*. ACM, 349–352.

[7] J. Chen, C. Wang, and J. Wang. 2015. Will you" reconsume" the near past? fast prediction on short-term reconsumption behaviors. In *Conference on Artificial Intelligence*.

[8] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 714–722.

[9] H. Dai, Y. Wang, R. Trivedi, and L. Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[10] E. Frolov and I. Oseledets. 2017. Tensor methods and recommender systems. *Interdisciplinary Reviews: DataMining and Knowledge Discovery* (2017).

[11] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2019. Unsupervised Neural Generative Semantic Hashing. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 735–744.

[12] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2020. Content-Aware Neural Hashing for Cold-Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 971–980.

[13] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2020. Unsupervised Semantic Hashing with Pairwise Reconstruction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2009–2012.

[14] N. Hariri, B. Mobasher, and R. Burke. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*. 131–138.

[15] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based recommendations with recurrent neural networks. (2016).

[16] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] M. Kaminskas, F. Ricci, and M. Schedl. 2013. Location-aware music recommendation using auto-tagging and hybrid matching. In *Conference on Recommender systems*. ACM, 17–24.

[18] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Y. J. Ko, L. Maystre, and M. Grossglauser. 2016. Collaborative Recurrent Neural Networks for Dynamic Recommender Systems. In *ACML*, Vol. 63.

[20] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37.

[21] S. Kumar, X. Zhang, and J. Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *International Conference on Knowledge Discovery & Data Mining*. ACM, 1269–1278.

[22] R. Mehrotra, J. McInerney, H. Bouchard, M. Lalmas, and F. Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *International Conference on Information and Knowledge Management*. ACM, 2243–2251.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[24] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy. 2019. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature Human Behaviour* 3, 3 (2019), 230.

[25] M. J. Pazzani and D. Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[26] T. F. Pettijohn, G. M. Williams, and T. C. Carter. 2010. Music for the seasons: seasonal music preferences in college students. *Current Psychology* 29, 4 (2010), 328–345.

[27] M. Quadrana, P. Cremonesi, and D. Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[28] S. Raza and C. Ding. 2019. Progress in context-aware recommender systems—an overview. *Computer Science Review* 31 (2019), 84–97.

[29] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *Conference on Artificial Intelligence*, Vol. 33. 4806–4813.

[30] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi. 2018. Current challenges and visions in music recommender systems research. *Journal of Multimedia Information Retrieval* (2018), 95–116.

[31] G. Shani, D. Heckerman, and R. I. Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* (2005), 1265–1295.

[32] D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, R. Henao, and L. Carin. 2018. NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2041–2050.

[33] A. Vall, M. Quadrana, M. Schedl, G. Widmer, and P. Cremonesi. 2017. The Importance of Song Context in Music Playlists.. In *RecSys Posters*.

[34] S. Wang, L. Cao, and Y. Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).

[35] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing. 2017. Recurrent recommender networks. In *International conference on web search and data mining*. ACM, 495–503.

[36] S. Zhang, Y. Tay, L. Yao, and A. Sun. 2018. Next Item Recommendation with Self-Attention. *ArXiv* abs/1808.06414 (2018).

[37] S. Zhang, L. Yao, A. Sun, and Y. Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019).

[38] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Conference on Artificial Intelligence*.

[39] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 717–726.

[40] A. Zimdars, D. M. Chickering, and C. Meek. 2001. Using temporal data for making recommendations. In *Conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 580–588.