Predicting Sequential User Behaviour with Session-Based Recurrent Neural Networks

Our Approach to the 2019 WSDM Cup Sequential Skip Prediction Challenge

Olivier Jeunen
University of Antwerp, Belgium
olivier.jeunen@uantwerp.be

Bart Goethals
University of Antwerp, Belgium
Monash University, Australia
bart.goethals@uantwerp.be

ABSTRACT

Online music streaming services have known a colossal growth in recent years. The freedom of picking whichever song a user wants to listen to, combined with the ease of letting the system generate personalised playlists to fit a user's specific taste, are only some of the aspects that led to this surge in popularity. As a consequence, a significant amount of research has lately focused on optimising the personalisation and recommendation processes in this musical context. Most often, these systems focus on positive feedback, recommending items similar to those we know the user has listened to before. However, much richer feedback signals can be considered. If, for example, a user explicitly skips a given song, we can infer that other songs were preferred at that given time. As a consequence, accurately predicting how users will interact with a given recommendation is a valuable feat. To this end, we present a novel approach for predicting sequential user behaviour. Our proposed method comprises a modular and scalable pipeline, a minimum of feature engineering and a single recurrent neural network architecture trained with a weighted loss function.

We evaluate our methodology with data provided by Spotify, in the Sequential Skip Prediction Challenge that was part of the 2019 WSDM Cup. In this challenge, our approach achieved the 5th position with the 'Adrem Data Lab' team.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Neural networks**; • **Information systems** → Personalisation; Recommender systems;

KEYWORDS

Classification, sequential prediction, recurrent neural networks, personalisation

ACM Reference format:
Olivier Jeunen and Bart Goethals. 2019. Predicting Sequential User Behaviour with Session-Based Recurrent Neural Networks. In *Proceedings of the 2019 WSDM Cup Workshop, Melbourne, Australia, February 15th 2019 (WSDM Cup '19),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

WSDM Cup '19, February 15th 2019, Melbourne, Australia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The rise of the world-wide-web has fundamentally changed how we consume content. News articles are mostly read online, traditional retail businesses are shifting towards e-commerce, movies are streamed instantly with the click of a button, and the same holds for music. The latter, however, differs from other domains in several key aspects, especially when considering how users interact with items [13]. One example is repeat consumption: it is not unlikely for a user to listen to a given song on multiple days in the same week, a phenomenon that would be considered rare in many other domains. Moreover, different contexts yield very different behaviour between the user and the item catalogue. Listening to an automatically generated playlist or a specifically chosen album, on a mobile or desktop device, in the middle of the day or night, … brings about rather diverse interaction patterns. Nevertheless, one common aspect in these scenarios is that users tend to listen to multiple songs in a row, generating so-called *sessions*. Throughout such a session, a user might listen to every song in its entirety, or skip songs at any given time. This latter action can stem from various underlying causes: (1) the user did not like the song, (2) the user did not like the song *in this specific context*, (3) the user saw the upcoming song, and preferred it over the current song, (4) the songs was unknown to the user and she was not in an exploratory mood, and many more options exist.

Although specific reasons vary, skipping a song will often indicate that a more appropriate song was available. With this in mind, there is a rich feedback signal to be exploited from such interaction data. If we can better understand what factors lead to users skipping or not skipping songs, and in turn are able to predict how users will interact with a given recommendation in a given context, this can incur significant benefit to the entity providing the recommendations.

In this paper, we present a novel approach for sequentially predicting how users will interact with future items throughout a given session. Our methodology comprises a modular and scalable pipeline, a minimum of feature engineering and a single recurrent neural network architecture[1]. We evaluate our proposed approach with data provided by Spotify, in the Sequential Skip Prediction Challenge that was part of the 2019 WSDM Cup [1]. In this challenge, our approach achieved the 5th position under the 'Adrem Data Lab' team. Throughout our work, we focus on the task of predicting whether a user will skip a given song, but our method is broadly applicable for sequentially predicting user behaviour.

[1]We provide an open-source implementation of our work at https://github.com/olivierjeunen/sequential-skip-prediction.

2 PROBLEM DESCRIPTION

The fields of recommender systems and information retrieval have recently focused more and more on the music domain, as demonstrated in part by the 2018 WSDM and RecSys challenges [2, 3]. A large part of the research focuses on devising novel algorithms to tackle the recommendation and personalisation tasks [9, 12]. How users sequentially interact with catalogue items, however, is a much less studied problem [13]. Accurately predicting how users will interact with a possible recommendation, can allow the system generating recommendations to improve itself by avoiding those with a high probability to be skipped, and vice versa. In what follows, we describe the dataset and evaluation method that were adopted for the competition.

2.1 Data

The dataset used in the challenge contains information about listening sessions and user interactions, generated from Spotify's online music streaming service [1]. *Sessions* are defined as sequences of listened songs. For every session, content features for the specific tracks are provided, along with a given context. The provided interaction data denotes whether or at which point the various songs were skipped. Content features of a song include the duration, release year, key, and more. Context data of a session includes the time of day, whether the user has a premium subscription, whether the *shuffle* flag is on, whether the user is listening to a playlist or an album, et cetera[2]. Sessions consist of anywhere between 10 and 20 songs, and nearly 4 million unique songs appear throughout the dataset. All data for training and testing was collected between July 15th and September 18th 2018. The full training set consists of roughly 130 million sessions, or 2 billion session-song pairs. When splitting the data according to the date the session was generated, we obtain 66 blocks with roughly 2 million sessions and 30 million session-song pairs each. The full test set consists of 31 million sessions, and 588 million session-song pairs.

2.2 Evaluation

All sessions are divided in half based on time. The first half of the session is labeled and includes contextual features as well as interaction data, whereas the second half only includes an ordered sequence of songs along with their corresponding metadata. Based on the preceding interactions in the first half of the session, the goal is to accurately predict whether the songs in the second half will be skipped. A prediction is a binary variable: 1 indicates the user will skip the song, 0 indicates the opposite. In order to put more weight on the first subsequent prediction and less weight on predictions further down the session, Mean Average Accuracy (MAA) is used as the primary evaluation metric. Equation 1 defines Average Accuracy (AA) for two given binary vectors $\mathbf{y}$ and $\mathbf{y}'$, respectively containing ground truth labels and predictions.

$$\mathrm{AA}(\mathbf{y}, \mathbf{y}') = \frac{1}{|\mathbf{y}|} \sum_{i=1}^{|\mathbf{y}|} A(i) \cdot \delta_{\mathbf{y}_i, \mathbf{y}'_i} \tag{1}$$

[2]We omit further detail of the different data fields due to space restrictions, but more information can be consulted on the challenge's web-page: https://www.crowdai.org/challenges/spotify-sequential-skip-prediction-challenge.

$A(i)$ denotes the accuracy at position i, and the Kronecker-delta $\delta_{\mathbf{y}_i, \mathbf{y}'_i}$ is a binary variable indicating whether the prediction at position i was correct. MAA is in turn defined as the mean AA for all predictions in the test set. The accuracy of the first prediction in a session is used as a secondary tie-breaking metric, which we will denote as First Prediction Accuracy (FPA).

3 METHODOLOGY

In what follows, we provide an overview of our approach. First, we present the general pipeline and framework we used to tackle the challenge in a modular and scalable way. Then, we introduce the relevant features we engineered from the training data. Our recurrent neural network architecture is described and motivated in Section 3.2. Section 3.3 introduces the weighted loss function we used as a differentiable proxy to indirectly optimise for the competition metric: MAA.

3.1 Preprocessing and Pipeline

Except for session or song identifiers, all available features were included in our model. Categorical features were one-hot encoded, and all features were standardised to have zero mean and unit variance. As the cardinality of the song identifiers approaches 4 million, it becomes problematic to include this feature as-is in the model. Nevertheless, which specific song the user listens to would certainly be a useful feature to predict whether it will be skipped. We engineered several features to capture some of the information that the identifiers otherwise would have represented: (1) the number of times the specific song appeared within the same session, (2) the number of unique songs in this session, and (3) the number of times the specific song appeared over all sessions (we log-normalised this feature in order to alleviate the long tail distribution [10]).

Let n be the maximal session length. With $k = \lfloor n/2 \rfloor$, we divide every input session into the first k tracks, and the last $n-k$. In doing this, we obtain two 3-dimensional input matrices: $\mathbf{X}_h \in \mathbb{R}^{m \times k \times f}$ and $\mathbf{X}_f \in \mathbb{R}^{m \times (n-k) \times f'}$. Here, m represents the number of input sessions, f is the total number of features, and f' denotes the number of features relating to the individual tracks. $\mathbf{X}_h$ represents all information we have about the tracks in the first half of the session, including contextual and interaction-based features. $\mathbf{X}_f$ only includes content- and track-based features about songs in the second half, for which a prediction needs to be made. As session lengths can vary, shorter sessions are padded with zeroes to match the size of the longest ones. Historical tracks $\mathbf{X}_h$ are padded before the track sequence, and future tracks $\mathbf{X}_f$ are padded after the track sequence.

Due to the size of the dataset, fitting it entirely in memory would require exorbitant amounts of resources. Moreover, as we are dealing with temporal data, new songs start to appear throughout the data, songs' popularities change over time, et cetera. This phenomenon, widely known as concept drift, can heavily influence the performance of learning algorithms, if not dealt with properly [15]. Because of those reasons, we split the dataset based on the date of data collection. We obtain 66 independent parts, every part corresponding to a single day of data collection. For every day, we only use half of the available training data. On these daily blocks of training data, we train models with 5-fold cross-validation [8].

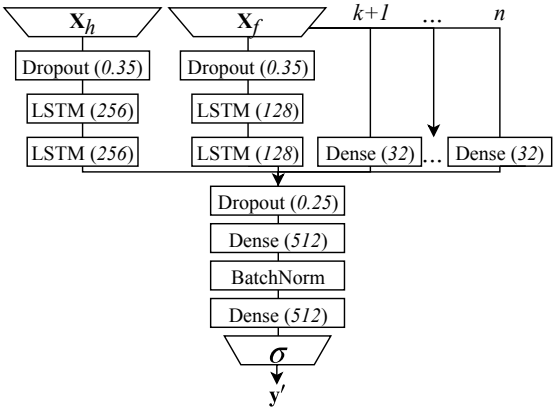

Figure 1: A schematic representation of the recurrent neural network (RNN)'s architecture. $\mathbf{X}_h$ is a matrix containing the tracks in the first half of the session as rows, with content, context and interaction features as columns. Analogously, $\mathbf{X}_f$ holds all tracks in the second half of the session, with their corresponding content- and track-features. Every track in $\mathbf{X}_f$ corresponds to a prediction in $\mathbf{y}'$.

Final predictions for the samples in the test set are computed as the geometric mean of the predictions generated by models trained on different folds for the same date as the test samples.

As every model is trained on less than 1% of all available training data, our approach is highly data-efficient. Furthermore, all daily models are entirely independent from one another, rendering model training an embarrassingly parallel task. This makes our approach easily distributable and highly scalable.

3.2 Network Architecture

Figure 1 shows a schematic representation of the recurrent neural network's (RNN) architecture. We employed dropout to prevent overfitting [14], as well as batch normalisation to further regularise the network [6]. Input sequences are modelled using doubly stacked long short-term memory (LSTM) layers [5]. Every future track, represented by a row in $\mathbf{X}_f$, is connected to a dense layers. The outputs of these LSTM and dense layers are concatenated and fed into the tail of the network. All dense layers are followed by exponential linear units (ELUs) [4]. In our architecture, information about tracks $i+1$ to n is available for the network to generate a prediction about whether track i will be skipped. As many user interfaces allow for the user to see upcoming tracks, we consider this to be realistic and valuable.

We generate $n-k$ predictions for every session, including those that consist of less than n tracks. To mitigate this issue, we pad the label vector $\mathbf{y}$ with zeroes, analogously to the padding performed with $\mathbf{X}_f$. However, note that these padded labels and predictions are not considered when evaluating the model using Equation 1. Network weights are learned through AMSGrad [11], which optimises a weighted variant of binary cross-entropy as loss function.

3.3 Weighted Binary Cross-Entropy

Binary cross-entropy or log-loss is a widely known and commonly used loss function for models that need differentiable objectives. It is often used as a proxy for optimising accuracy, as it penalises

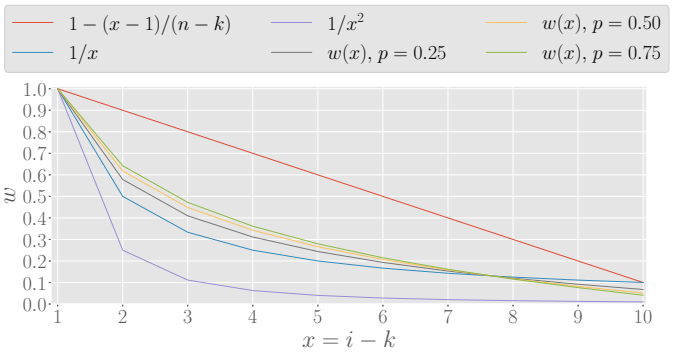

Figure 2: Weights per position for alternative weight functions, and varying values of p. Every plotted value is divided by the maximum value for that function, i.e. $f(1)$, to allow for more intuitive comparison. ($n=20$)

incorrect predictions based on how confident those predictions are. From the MAA metric shown in Equation 1, it is clear that predictions closer to the middle of the session are more important than those nearing the end. This phenomenon motivated us to use a weighted variant of binary cross-entropy as a loss function, with weights corresponding to the predictions' importance in Equation 1.

A prediction $\mathbf{y}'_i$ at position i has a direct influence on $A(i)$, maximally increasing it by $1/(i-k)$. Analogously, the same prediction $\mathbf{y}'_i$ influences all accuracies further down the session, i.e. $A(j)$ for all $j \in (i, n]$, maximally increasing it by $1/(j-k)$. However, whether $A(j)$ is taken into account for $\mathrm{AA}(\mathbf{y}, \mathbf{y}')$ is entirely dependent on whether the prediction at position j is correct, i.e. $\delta_{\mathbf{y}_j, \mathbf{y}'_j}$. If we assume that whether the prediction at position j is independent of whether the prediction at position i is correct, and we assume that the probability of a prediction being correct is fixed, we can analytically formulate the weights for every position. We will represent this probability of a prediction being correct as p. Equation 2 shows the formula for computing the weight for a prediction at position i.

$$w(i) = \frac{1}{i-k} + \sum_{j=i+1}^{n} \frac{p}{j-k} \tag{2}$$

Figure 2 visualises different weight functions, and $w(i)$ for different values of p. It is clear to see from both Equation 2 and Figure 2 that $w(i)$ converges to $1/(i-k)$ as p approaches 0.

4 EXPERIMENTAL RESULTS

To experimentally validate the value of the individual components of our approach, we report results for mean MAA and FPA on a held-out test set. Our local evaluation approach follows the same structure as the pipeline laid out in Section 3.1: Table 1 shows results on a held-out test set, after 3-fold cross validation on the first 5 available log-files for the first date. We only report results from local evaluation on data from the first date, as we noticed a strong correlation in terms of ratios in metrics among competing algorithms between our local evaluation and the leaderboard results from the test-set. 'Random' is a baseline that uniformly at random predicts whether a track will be skipped or not. 'RNN' represents the network shown in Figure 1, without the weighted loss function following Equation 2 or the features engineered from the

Table 1: Mean MAA and FPA on a held-out test set, after 3-fold cross validation. Models were trained on 50% of the available data for the first date in the training data, similar to the approach we followed when generating predictions for the test set. 'w-RNN$_f$', the model variant including custom features and a weighted loss function, outperforms all other variants. However, the highest performance gains can be attributed to the track-based features.

	Random	RNN	w-RNN	RNN$_f$	w-RNN$_f$
MAA	0.334	0.583	0.584	0.607	**0.608**
FPA	0.500	0.782	0.783	0.791	**0.792**

track-identifiers, as described in Section 3.1. 'RNN' incorporates the available data as-is as its input. 'w-RNN' includes the weighted loss function ($p = 0.5$) but not the features, 'RNN$_f$' includes the features but not the weighted loss function. Finally, 'w-RNN$_f$' is our final algorithm that incorporates both. From Table 1, it is clear to see that the 'w-RNN$_f$' model variant outperforms all others. While the use of the weighted loss function consistently improves the evaluation metrics, its gains seem marginal compared to the improvement caused by including custom track-features in the model.

On the competitive test set, we finally scored an MAA of **0.613** and an FPA of **0.794**. The improvement of results on the competitive test set in comparison with our local evaluation can be explained by variance in the data[3], and a small boost by model averaging over 5 folds for the competition instead of 3 for our local evaluation procedure. If we can trust our local evaluation procedure and assume a correlation with performance on the private test set, even our simplest model variant 'RNN' would have achieved a spot among the top 10 highest-scoring contestants.

5 CONCLUSION

We have motivated the need for algorithms that can sequentially predict how users will interact with items they are presented with. In the music domain, where users tend to consume multiple items in sessions and explicit 'skip'-interactions are logged, the usefulness of such a predictor is most apparent. To this end, we presented a novel approach for predicting sequential user behaviour. Our methodology is based on a single RNN architecture that uses a minimum of engineered features. We propose a data-efficient, modular and scalable pipeline that allows for our method to be used with real-world, industrial-sized datasets. Throughout our work, we focused on the task of predicting whether a user will skip a given song. Nevertheless, our methodology is broadly applicable for sequentially predicting user behaviour.

The efficacy and efficiency of our approach was validated with data provided by Spotify, in the Sequential Skip Prediction Challenge that was part of the 2019 WSDM Cup [1]. In this challenge, we achieved 5th position as the 'Adrem Data Lab' team. However, in non-competitive environments, model training and evaluation should adhere to strict temporal constraints: user-item interactions occurring in training data should have always happened before those occurring in test data for an offline evaluation procedure to provide an accurate estimate of online performance [7].

[3]Local evaluation results with data generated on different dates, generally fluctuated with ~ 0.05 as well.

5.1 Future Work

Although our approach attained promising results, we list several valuable extensions as future research directions: (1) For sequence data, RNNs are often a go-to approach. The use of alternative neural architectures such as Convolutional Neural Networks (CNNs), Capsule Networks or others might bring significant gains. (2) More classical classification models such as Gradient-Boosted Decision Trees (GBDT) have been shown to be highly competitive in classification tasks as well. (3) Due to the high cardinality of the track-identifiers, we were unable to explicitly incorporate these into the model. As Table 1 shows, track-based features entail high performance gains for the model, further research in this direction could certainly prove to be beneficial.

ACKNOWLEDGMENTS

We thank the organisers of the 2019 WSDM Cup, as well as Spotify for providing the data that facilitated this competition. Furthermore, we thank the Flemish Supercomputer Centre for providing us with the computational resources needed to accomplish this work.

REFERENCES

[1] B. Brost, R. Mehrotra, and T. Jehan. 2019. The Music Streaming Sessions Dataset. In *Proc. of the 2019 Web Conference*. ACM.
[2] C. Chen, P. Lamere, M. Schedl, and H. Zamani. 2018. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *Proc. of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, 527–528.
[3] Y. Chen, X. Xie, S. Lin, and A. Chiu. 2018. WSDM Cup 2018: Music Recommendation and Churn Prediction. In *Proc. of the 11th ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, 8–9.
[4] D. Clevert, T. Unterthiner, and S. Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proc. of the 4th International Conference on Learning Representations (ICLR'16)*. 14.
[5] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[6] S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd International Conference on Machine Learning (ICML'15)*. 448–456.
[7] O. Jeunen, K. Verstrepen, and B. Goethals. 2018. Fair Offline Evaluation Methodologies for Implicit-feedback Recommender Systems with MNAR Data. In *Proc. of the REVEAL 18 Workshop on Offline Evaluation for Recommender Systems (RecSys '18)*.
[8] R. Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the 4th International Joint Conference on Artificial Intelligence (IJCAI '95)*, Vol. 14. 1137–1145.
[9] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra. 2018. Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits. In *Proc. of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, 31–39.
[10] Y. Park and A. Tuzhilin. 2008. The Long Tail of Recommender Systems and How to Leverage It. In *Proc. of the 2nd ACM Conference on Recommender Systems (RecSys '08)*. ACM, 11–18.
[11] S. J. Reddi, S. Kale, and S. Kumar. 2018. On the Convergence of Adam and Beyond. In *Proc. of the 6th International Conference on Learning Representations (ICLR'18)*. 23.
[12] N. Sachdeva, K. Gupta, and V. Pudi. 2018. Attentive Neural Architecture Incorporating Song Features for Music Recommendation. In *Proc. of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, 417–421.
[13] M. Schedl, H. Zamani, C. Chen, Y. Deldjoo, and M. Elahi. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval* 7, 2 (01 Jun 2018), 95–116.
[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[15] G. Widmer and M. Kubat. 1996. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning* 23, 1 (01 Apr 1996), 69–101.