

# Spotify User Skip Prediction Using LSTM

Ana Escoto-Cardenas

**Abstract**—Music consumption habits of people have changed over the years. As the public has widely migrated to use music streaming services, it gives the service platforms such as Spotify, Apple Music and Tidal, a massive incentive to predict whether a particular user will skip a particular song based on their previous actions. This can be very helpful in creating a better user experience and hence, spend more time on that platform. One of the challenges for these platforms is to determine what kind of music the user would like to hear. The skip button is one powerful indicator as to whether the user enjoys the music or not. Users are free to abandon whatever song they are listening. Thus, for this project, I will use Recurrent Neural Networks (RNNs) to predict whether a user will skip a particular song given the users previous information during a listening session along with the acoustic features of previous tracks.

## I. INTRODUCTION

One of the biggest challenges facing streaming services is the ability to accurately recommend content that their customers will have a positive reaction to. Being able to provide customers with content that evokes a positive reaction leads to a better customer experience, thus increasing customer retention. This paper will discuss the task of predicting whether a user will skip the last track in their listening session based on their previous listening history and track features.

The problem discussed in this paper can be yield as a sequential problem due to the fact that user behavior per session is temporal information. Recurrent Neural Networks with Long Short Term Memory (LSTM) [1] have been shown to deliver high accuracy in sequence modeling techniques such as Natural Language Processing (NLP) and Predictive Process Motoring problems. The paper introduces LSTM architectures for predicting whether a user will skip the last track in their listening session using i) Historical session behavior ii) Meta information of previous tracks listened during session.

## II. RELATED WORK

The inspiration behind this project was the Spotify Sequential Skip Challenge that was conducted in 2019. A range of works have discussed approaches to this task, one of the most important ones was the “Thank you, Next” paper that was written by Alex Hurtado, Markie Wagner and Surabhi Mundada [3]. They used Gradient Boosted Tree as the preliminary baseline method of building onto previous data which in this case are the previous tracks. They used LSTM (Long-Short-Term-Memory) type of Recurrent Neural Networks. This paper massively influenced and clarified what measures and ideas needed to be implemented since my paper was very closely mapped according to their approach.

Another paper that proved fairly important to grasp the ideas of this project was the paper called “Predicting Sequential user Behavior with Session-Based Recurrent Neural Networks” by Oliver Jeunen and Bart Goethals from Belgium. In this paper, they presented a novel approach for sequentially predicting how users will interact with future items throughout a given session. They used feature engineering and a single recurrent neural network architecture. This paper proved important for us to understand temporal variables and how they work.

## III. DATA

my overall data-set consists of two separate data sets: - *User Log* which describes user sessions. my data-set contains 10,000 different sessions ranging from lengths of 10 tracks listened to 20 tracks listened. Each session has a unique “Session ID” and contains general information about the Spotify user such as whether they are a premium customer, what time of day they are listening to the track, and the action the user took to end the last track (i.e. skipped forward, listened to it’s entirety, or skipped back). Lastly, the dataset contains a “track id” for each session position which will allow us to link the two separate datasets to create a master data-set for my task.

-*Track Meta* which describes the meta of each unique track\_id. my data-set contains 50,704 different tracks along with their unique characteristics. These characteristics are features such as duration, release year, U.S. popularity, as well as acoustic features such as flatness, energy, tempo, and time signature. All 50,704 unique tracks appear in my User Log data-set described above.

The full data-set provided by Spotify consists of 31 million sessions, and 588 million tracks. Due to the size of that data-set, using a data-set that big will be nearly impossible for my personal machines to process that amount of information. Due to this limitation, I have limited the size of my data-set to 10,000 sessions with a total of 167,880 data-points and track meta size of 50,704 tracks.

TABLE I  
USER LOG FEATURES

Feature	Description
<i>session_id</i>	Primary key to differentiate different sessions
<i>session_position</i>	Ranges from 1-20
<i>session_length</i>	Ranges from 10-20
<i>track_id</i>	foreign key to link to track meta data-set
<i>context_type</i>	Context of listening session i.e. Radio, Personalized Playlist, Charts
<i>context_switch</i>	User switched contexts from previous session position
<i>no_pause_before_play</i>	User did not pause streaming before current session position
<i>short_pause_before_play</i>	User had short pause streaming before current session position
<i>long_pause_before_play</i>	User had long pause streaming before current session position
<i>hist_user_behavior</i>	User skipped forward or back before current session position
<i>shuffle</i>	User is in shuffle mode
<i>hmy_of_day</i>	Time in which user is in current session position
<i>Date</i>	Day in which user is in current session position
<i>Premium</i>	User is a premium customer
<i>user_behavior_start</i>	Why current session position started i.e. trackdone, fwdbtn,backbtn, etc.
<i>user_behavior_ended</i>	Why current session position started i.e. trackdone, fwdbtn,logout, etc.

TABLE II  
TRACK META FEATURES

Features	Features cont.
<i>track_id</i>	<i>mode</i>
<i>duration</i>	<i>speechiness</i>
<i>release_year</i>	<i>tempo</i>
<i>us_popularity_estimate</i>	<i>time_signature</i>
<i>acousticness</i>	<i>valence</i>
<i>beat_strength</i>	<i>accoustic_vector_0</i>
<i>bounciness</i>	<i>accoustic_vector_1</i>
<i>danceability</i>	<i>accoustic_vector_2</i>
<i>dyn_range_mean</i>	<i>accoustic_vector_3</i>
<i>energy</i>	<i>accoustic_vector_4</i>
<i>flatness</i>	<i>accoustic_vector_5</i>
<i>instrumentalness</i>	<i>accoustic_vector_6</i>
<i>key</i>	<i>accoustic_vector_7</i>
<i>liveness</i>	

### A. Data Preprocessing

1) *Merging Data-sets*: The first part of my Data Preprocessing step was to combine both of my User Log and Track Meta data-sets using their linking feature "track\_id". After combining these two data-sets, my final data-set ended up having a total of 167,880 rows and 50 columns.

2) *Categorical Variables*: Next was to tackle my categorical features. All 7 categorical features I re One-hot Encoded due to the fact that they had no ordinal characteristics. After One-hot Encoding my 7 categorical features, I had a complete data-set of 167,880 rows and 69 columns.

3) *Feature Scaling*: Features from my Track Meta ranged from decimal values to values as high as 100, therefore, I decided to scale them to values within [0,1].

### B. Response Variable

Due to the fact that my sessions vary in length, the last track of different sessions vary in position. Meaning, my response variable (y) can be in position 10 for a session of length 10 and position 20 for a session of length 20. my response variable is the Boolean variable "not\_skipped", a 0 indicates the track was skipped and a 1 indicates a track was not skipped. Every session position contains this response variable. However, for my task, I am only interested in predicting "not\_skipped" response variable for the last session position. my data-set does not seem to be severely unbalanced. Figures 1 & 2 show the balance of both my training and testing set, therefore no re-balancing techniques I re performed

TABLE III  
RESPONSE VARIABLE

Response Variable	Description
<i>not_skipped</i>	Boolean response variable 0: Skipped 1: Not skipped

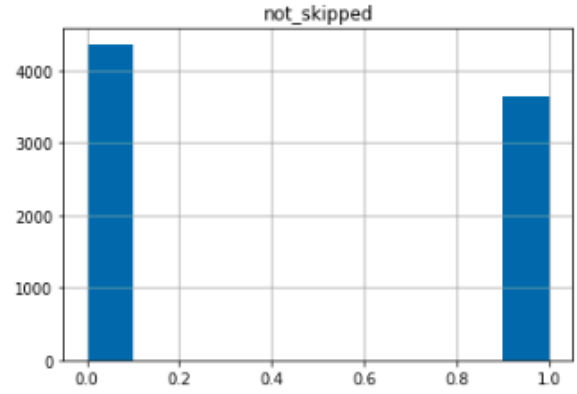


Fig. 1. Train Balance

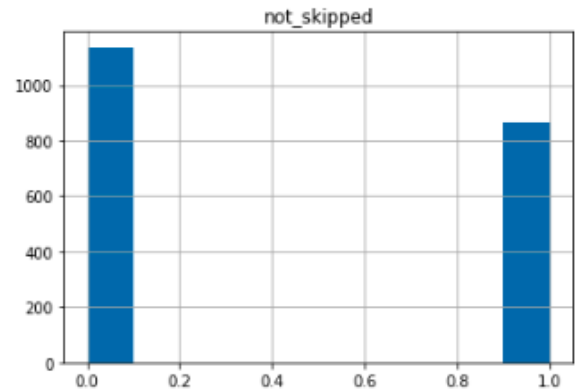


Fig. 2. Test Balance

### C. Train and Test Split

I split my data into 80% training and 20% testing. This allowed us to use 8,000 sessions for training and 2,000 sessions for testing. Training had a total of 126,274 rows and Testing had a total of 31,606.

## IV. METHODS

This section will explain the inputs as well as the neural network architecture of my model. Overall my network consists of 4 different networks with 4 different inputs: Previous Session Logs, Previous Session Track Meta, Previous Skip Sequence, and Last Session Position Track Meta. All four of these inputs will have their own network and I will concatenate them to predict whether the last session position will be skipped.

### A. Inputs

1) *Previous Session Logs*: This input is a (19, 35) array that includes all session information leading up to my last session position. Information in this input includes: whether user is premium, session length, hist\_user\_behavior, user\_behavior\_start, etc. Since my session lengths vary in size, I have padded my sequences. The max length for session is 20, therefore, the max length of session positions leading up to my final track is 19. Therefore, I have padded my sequences to 19 as seen in the middle variable of my array.

2) *Previous Session Positions Track Meta*: Similarly to my Previous Session Logs, this input contains all information of tracks listened in session positions < Last Position. This input is a (19,31) array that has been padded to the max length of session positions leading up to my final track which is 19.

3) *Previous Session Positions Skip Sequence*: This input is a sequence of 0s and 1s depicting whether tracks were skipped in previous session positions, this input is a (19,1) array. Since I have varying session lengths, I decided to pad the beginning of the sequence with the value "9". I decided to pad with a value other than 0 due to the fact that in my data, the number 0 means "skipped" and padding my sequences with 0 could alter the performance.

$$x_1, x_2, \dots, x_{10}, x_{11}, \dots, x_{19}$$

4) *Last Session Position Track Meta*: Lastly, my fourth input contains all meta information of the track being played in the last session. This is the track for which I am ultimately trying to predict whether the user will skip or not. This is the only input in my model that does not contain any temporal variables.

### B. Network

My architecture consisted of 3 different LSTMs for my inputs containing temporal variables: Previous Session Logs, Previous Session Positions Track Meta, Previous Session Positions Skip Sequence. I also implemented a Dense Neural Network for my Last Session Position Track Meta input.

All four of these networks were concatenated to obtain my final prediction. First, Previous Session Positions Track Meta network and Last Session Position Track Meta network were concatenated. Next, Previous Session Logs and Previous Session Positions Skip Sequence were also concatenated. Lastly, I concatenated both merged networks into a Dense layer. All networks had varying dropouts ranging between 0.4 and 0.5. Further details on each individual network can be found below, as well as in Figure 7 showing my complete network.

1) *Previous Session Logs LSTM*: This model is composed of 2 LSTM layers with a 0.4 Dropout, ReLU activation, and 128 Nodes.

2) *Previous Session Positions Skip Sequence LSTM*: This model is composed of 2 LSTM layers with a 0.4 Dropout, ReLU activation, and 264 Nodes.

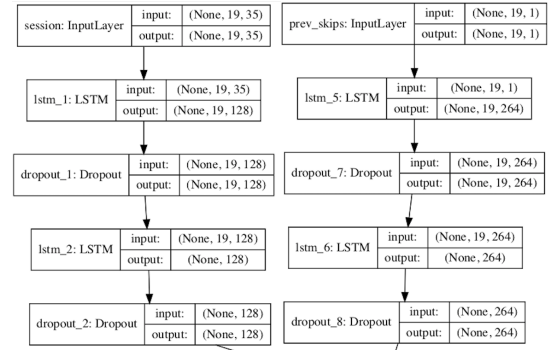


Fig. 3. LSTM 1

Fig. 4. LSTM 2

3) *Previous Session Positions Track Meta LSTM*: This model is composed of 2 LSTM layers with a 0.4 Dropout, ReLU activation and 128 Nodes.

4) *Last Session Position Track Meta NN*: This model is composed of 2 Dense NN with a 0.4 Dropout, ReLU activation, and 64 nodes.

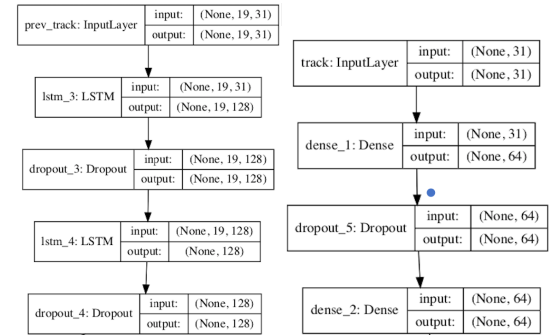


Fig. 5. LSTM 3

Fig. 6. Dense Network

For the last layer of my model, I decided to go with a Dense Layer with an output of 2 and a "sigmoid" activation function. I chose these two parameters due to the fact that my response variable is binary. Before this last Dense layer, I had a Dropout layer with parameter 0.4 in order to avoid over-fitting.

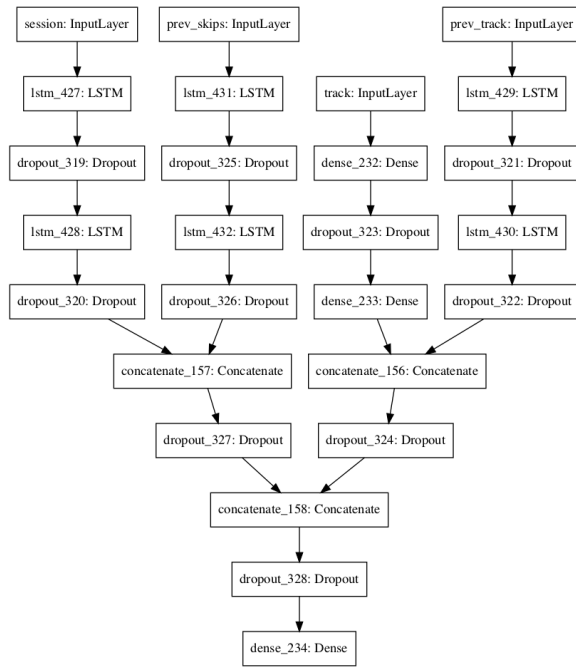


Fig. 7. Complete Network

## V. EXPERIMENTS

Trial and error was required to hyper-tune my model's parameters. Ultimately the parameters that proved to be the most optimal are the following:

- batch\_size = 848
- epochs = 100
- optimizer = 'adam'
- loss = 'binary\_crossentropy'

Although my epochs was set at 100, my model stopped training at epoch 59 due to Early Stopping. After Early stopping, my model reverted back to its best epoch lights which I found to be in epoch 34. In terms of initialization, the optimal Light initialization is random uniform and optimal bias initialization is a constant of 0.1.

The parameters that I noticed had the most positive effect on my model was the batch size and the optimizer. With optimizers such as RMSprop, or model only achieved an accuracy of 54%.

Another experiment conducted was having one single input containing all information about my previous session positions, instead of breaking them up into 4 individual inputs. The one input was then fed to an LSTM with 3 layers. This was my initial approach to the task, however, doing this resulted in an accuracy score of 66% which is much lower than the accuracy obtained by the approach I ultimately Int with and discuss in this paper.

## VI. RESULTS

my model achieved an accuracy score of 76% for my training set and 75% for my testing set, as seen in Figure 10. Or model has a precision of 0.77 and an F1-Score of 0.78 for response variable 0 (not skipped) and 0.70 F1-Score and

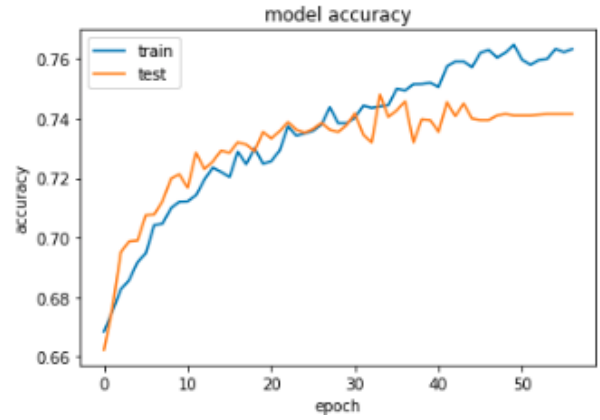


Fig. 8. Model Accuracy

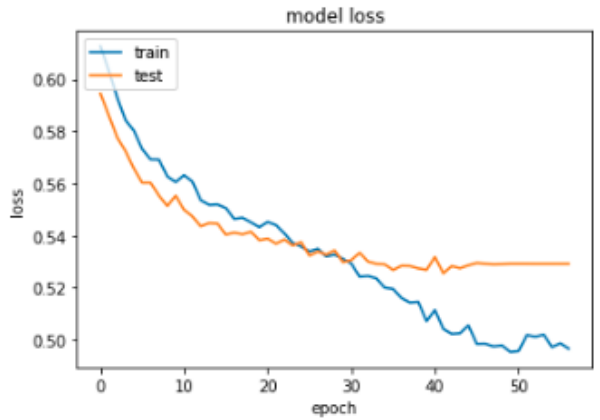


Fig. 9. Model Loss

0.72 precision score for response variable 1 (skipped). This is a significant improvement from my baseline of simply predicting 0 every time, which would yield an Accuracy Score of 55%. It is also a significant increase from my initial approach of only having 1 input and 1 LSTM, which yield an Accuracy Score of 66%.

Classification Report:				
	precision	recall	f1-score	support
0	0.77	0.79	0.78	1136
1	0.72	0.68	0.70	864
accuracy			0.75	2000
macro avg	0.74	0.74	0.74	2000
weighted avg	0.74	0.75	0.74	2000

Fig. 10. Confusion Matrix

## VII. CONCLUSIONS

The model described above has shown significant improvement from the baseline model (0.75 vs. 0.55). I am able to achieve these results by constructing 4 different models for my different inputs found in my data-set and merging them in order to provide a prediction. For future work, I would

like to tackle the task of predicting whether a user will skip multiple session positions, not just the last one. This type of task will allow us to experiment with NLP techniques such as Seq2Seq encoder/decoder frameworks.

#### REFERENCES

- [1] Charles Tremlett. 2019. Preliminary Investigation of Spotify Sequential Skip Prediction Challenge. WSDM (2019).
- [2] Sainath Adapa. 2019. Sequential modeling of Sessions using Recurrent Neural Networks for Skip Prediction.
- [3] Hurtado, A.F., Wagner, M., Mundada, S. (2019). Thank you, Next: Using NLP Techniques to Predict Song Skips on Spotify based on Sequential User and Acoustic Data.
- [4] Chang, S., Lee, S., Lee, K. (2019) Sequential Skip Prediction with Few-shot in Streamed Music Contents.