Spotify User Skip Prediction Using LSTM

Ana Escoto-Cardenas and Abdul Rahman

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, we 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 viewed 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 were 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 our 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

Our overall data-set consists of two separate data sets: -*User Log* which describes user sessions. Our 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 our task.

-*Track Meta* which describes the meta of each unique track_id. Our 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 our 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 our personal machines to process that amount of information. Due to this limitation, we have limited the size of our 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**
session_id	Primary key to differentiate different sessions
session_position	Ranges from 1-20
session_length	Ranges from 10-20
track_id	foreign key to link to track meta data-set
context_type	Context of listening session i.e. Radio, Personalized Playlist, Charts
context_switch	User switched contexts from previous session position
no_pause_before_play	User did not pause streaming before current session position
short_pause_before_play	User had short pause streaming before current session position
long_pause_before_play	User had long pause streaming before current session position
hist_user_behavior	User skipped forward or back before current session position
shuffle	User is in shuffle mode
hour_of_day	Time in which user is in current session position
Date	Day in which user is in current session position
Premium	User is a premium customer
user_behavior_start	Why current session position started i.e. trackdone, fwdbtn,backbtn, etc.
user_behavior_ended	Why current session position started i.e. trackdone, fwdbtn,logout, etc.

TABLE II
TRACK META FEATURES

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

A. Data Preprocessing

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

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

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

B. Response Variable

Due to the fact that our sessions vary in length, the last track of different sessions vary in position. Meaning, our response variable (y) can be in position 10 for a session of length 10 and position 20 for a session of length 20. Our 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 our task, we are only interested in predicting "not_skipped" response variable for the last session position. Our data-set does not seem to be severely unbalanced. Figures 1 & 2 show the balance of both our training and testing set, therefore no re-balancing techniques were performed

TABLE III
RESPONSE VARIABLE

Response Variable	**Description**
not_skipped	Boolean response variable 0: Skipped 1: Not skipped

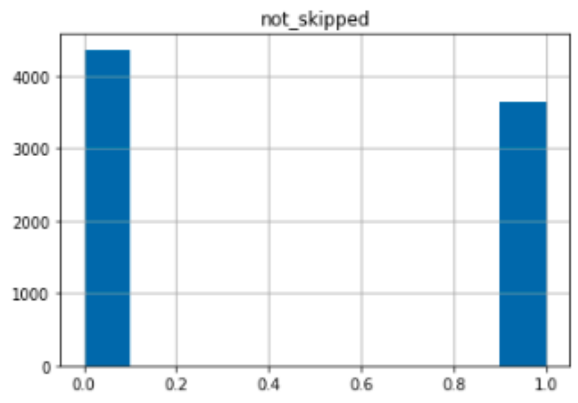


Fig. 1. Train Balance

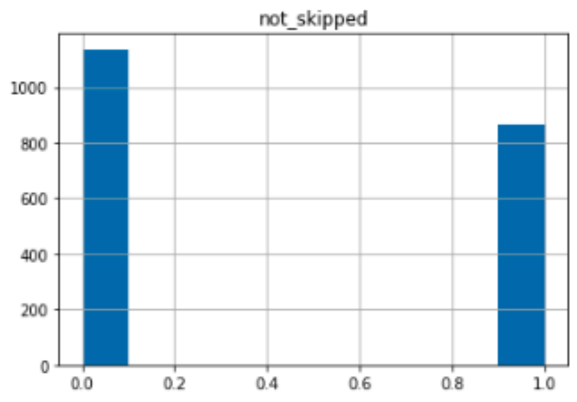


Fig. 2. Test Balance

C. Train and Test Split

We split our data into 80% training and 20% testing. This allowed us to use 8,000 sessions for training and 2,000 session 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 our model. Overall our 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 we 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 our last session position. Information in this input include: whether user is premium, session legth, hist_user_behavior, user_behavior_start, etc. Since our session lengths vary is size, we have padded our sequences. The max length for session is 20, therefore, the max length of session positions leading up to our final track is 19. Therefore, we have padded our sequences to 19 as seen in the middle variable of our array.

2) Previous Session Positions Track Meta: Similarly to our 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 our 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 we have varying session lengths, we decided to pad the beginning of the sequence with the value "9". We decided to pad with a value other than 0 due to the fact that in our data, the number 0 means "skipped" and padding our sequences with 0 could alter the performance.

$$x_1, x_2, ..., x_{10}, x_{11}, ..., x_{19}$$

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

B. Network

Our architecture consisted of 3 different LSTMs for our inputs containing temporal variables: Previous Session Logs, Previous Session Positions Track Meta, Previous Session Positions Skip Sequence. We also implemented a Dense Neural Network for our Last Session Position Track Meta input. All four of these networks were concatenated to obtain our 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, we 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 our 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.

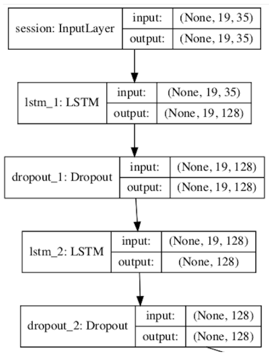


Fig. 3. LSTM 1

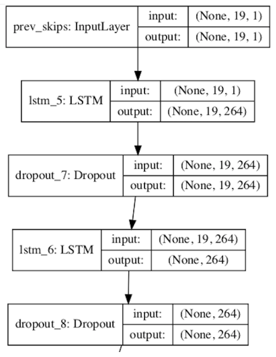


Fig. 4. LSTM 2

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.

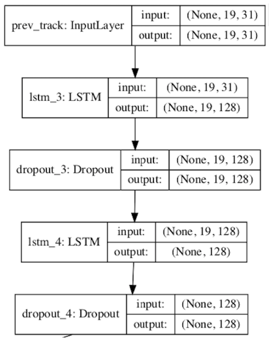


Fig. 5. LSTM 3

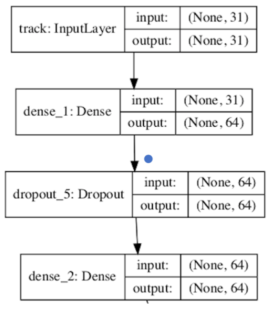


Fig. 6. Dense Network

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

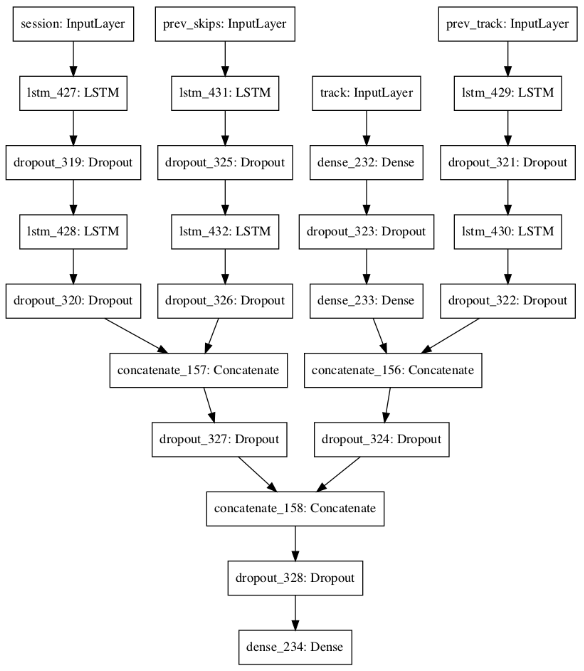


Fig. 7. Complete Network

V. Experiments

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

- batch_size = 848
- epochs = 100
- optimizer = 'adam'
- loss = 'binary_crossentropy'

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

The parameters that we noticed had the most positive effect on our 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 our 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 our 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 we ultimately went with and discuss in this paper.

VI. Results

Our model achieved an accuracy score of 76% for our training set and 75% for our 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 0.72 precision score for response variable 1 (skipped).This is a significant improvement from our baseline of simply predicting 0 every time, which would yield an Accuracy Score of 55%. It is also a significant increase from our initial approach of only having 1 input and 1 LSTM, which yield an Accuracy Score of 66%.

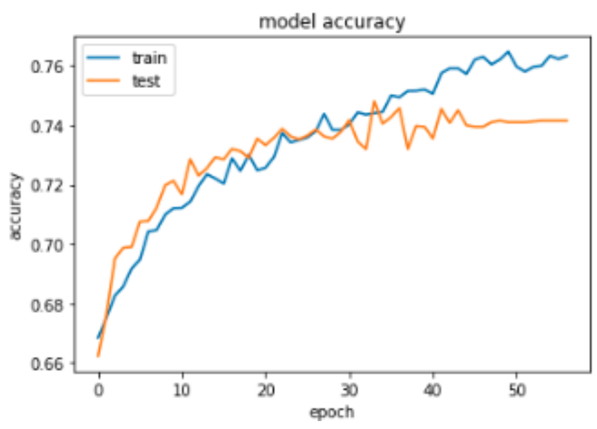


Fig. 8. Model Accuracy

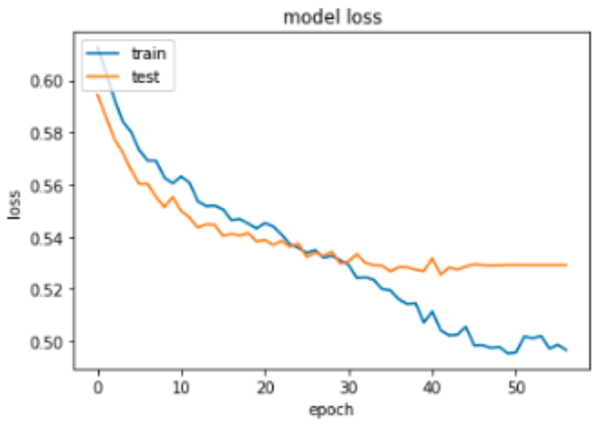
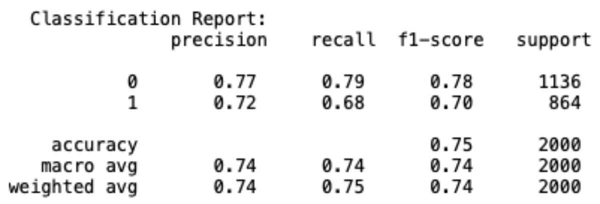


Fig. 9. Model Loss



```
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). We were able to achieve these results by constructing 4 different models for four different inputs found in our data-set and merging them in order to provide a prediction. For future work, we 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.

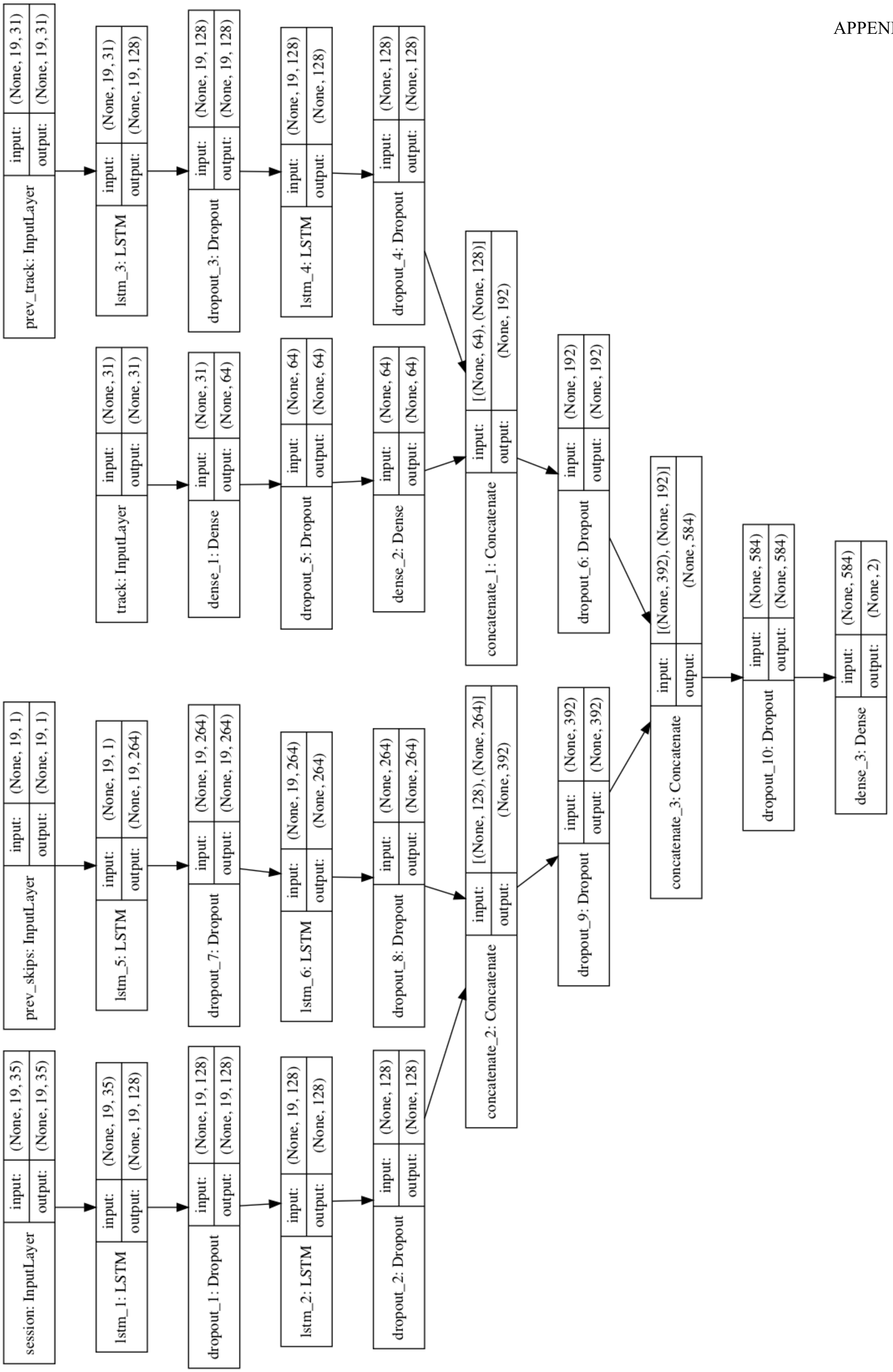
References

[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.


prev_track: InputLayer | input: (None, 19, 31) | output: (None, 19, 31)
lstm_3: LSTM | input: (None, 19, 31) | output: (None, 19, 128)
dropout_3: Dropout | input: (None, 19, 128) | output: (None, 19, 128)
lstm_4: LSTM | input: (None, 19, 128) | output: (None, 128)
dropout_4: Dropout | input: (None, 128) | output: (None, 128)
track: InputLayer | input: (None, 31) | output: (None, 31)
dense_1: Dense | input: (None, 31) | output: (None, 64)
dropout_5: Dropout | input: (None, 64) | output: (None, 64)
dense_2: Dense | input: (None, 64) | output: (None, 64)
concatenate_1: Concatenate | input: [(None, 64), (None, 128)] | output: (None, 192)
dropout_6: Dropout | input: (None, 192) | output: (None, 192)
prev_skips: InputLayer | input: (None, 19, 1) | output: (None, 19, 1)
lstm_5: LSTM | input: (None, 19, 1) | output: (None, 19, 264)
dropout_7: Dropout | input: (None, 19, 264) | output: (None, 19, 264)
lstm_6: LSTM | input: (None, 19, 264) | output: (None, 264)
dropout_8: Dropout | input: (None, 264) | output: (None, 264)
session: InputLayer | input: (None, 19, 35) | output: (None, 19, 35)
lstm_1: LSTM | input: (None, 19, 35) | output: (None, 19, 128)
dropout_1: Dropout | input: (None, 19, 128) | output: (None, 19, 128)
lstm_2: LSTM | input: (None, 19, 128) | output: (None, 128)
dropout_2: Dropout | input: (None, 128) | output: (None, 128)
concatenate_2: Concatenate | input: [(None, 128), (None, 264)] | output: (None, 392)
dropout_9: Dropout | input: (None, 392) | output: (None, 392)
concatenate_3: Concatenate | input: [(None, 392), (None, 192)] | output: (None, 584)
dropout_10: Dropout | input: (None, 584) | output: (None, 584)
dense_3: Dense | input: (None, 584) | output: (None, 2)