Using NLP Techniques to Predict Song Skips on Spotify based on Sequential User and Acoustic Data

1. Abstract

Music consumption habits have changed dramatically in the past decade with the rise of streaming services such as Spotify, Apple Music, and Tidal. Today, these services are ubiquitous. These music providers are also incentivized to provide songs that users like in order to create better user experience and increase time spent on their platform. Thus, an important challenge to music streaming is determining what kind of music the user would like to hear. The skip button is one feature instance on these music services that plays a large role in the users experience and helps identify what a user likes, as with this button, users are free to abandon songs as they choose. Thus, in this project, we will build a machine learning model that will predict if a user will skip a song or not given information about the user's previous actions during a listening session along with acoustic features of the previous songs.

2. Introduction

Music streaming platform like Spotify, Gaana, Apple Music and many other have been continuously increasing with time in the market because of continuous and increased in demand of listening music by user due to which there is a huge competition among Music provider. So, to retain and increase user on their platform, nowadays music provider continuously updates their platform by adding new features like recommending song that user might like, by showing song specific to user behaviour etc. But still, it quite challenging to provider about determining the kind of song user like or not and then recommending the right music on right time.

Machine learning has long been leveraged by provider to build recommender system in order to recommend music and this can be seen in various platform also like suggesting user new release, customizing playlist etc. However there has not been much research done on how a user interact with music over his listening session.

So, the aim of the project is to build a sequential skip prediction model using machine learning algorithm to predict if a user will skip the song or not based on the user interaction with previous songs and the song musical qualities in an individual music session in order to increase the user experience and the time spent on platform by recommending user the right song which the user wish to listen and continue with. One such user interaction is the skipping behavior which represent a powerful signal for our model to train and to knew from it about what user does and doesn't like. We have implemented simpler sequential-based models like GBTs, LSTMs, and Bi-LSTMS, as well as have applied NLP techniques to our task by leveraging and developing two versions of Transformers.

3. <u>Related Works</u>

The Spotify Sequential Skip prediction Challenge was a challenge open to the public, and so other groups have developed their own approaches to tackling this sequential prediction challenge. Chang et al. (Chang et al., 2019) experimented with concatenating the acoustic and behavior features of the first 10 tracks with the acoustic feature of the second 10 and sending this concatenation through several convolutional layers as well as a self-attention layer to output the predictions for tracks 10-20. They did not use an embedding layer, but directly sent in the concatenated feature vectors. This team explored both metrics and sequence learning methods, finding the sequence learning to be more successful. Beres et al. (Beres et al. ´, 2019) took a different approach to the challenge, instead choosing to use a Bi-LSTM to autoencoder as well as feature selection to create their model. This use of LSTMs prompted us to further explore the applications of encoders, leading us to eventually leverage other models typically used in NLP use cases." Preliminary Investigation of Spotify Sequential Skip Prediction Challenge" Charles Tremlett (Tremlett, 2019) simply used Gradient Boosted Trees and ensembled several of these trees to predict skips. For our novel methods, we were interested in leveraging the cutting-edge models in NLP that perform well on sequential prediction tasks. Ilya Sutskever et al. (Sutskever et al., 2014) first proposed the encoder decoder architecture for sequence-to-sequence learning in 2014. These sequence-to-sequence models map a fixed-length input sequence to a fixed-length target sequence. Further investigation led us to the Transformer model proposed by Vashwani et al (Vaswani et al., 2017). The Transformer revolutionized the world of NLP through the introduction of attention mechanisms, replacing recurrences and convolutions. Many of these NLP approaches for generating a sequence from a sequence use an encoder and/or decoder. In our project, we also use this encoder and decoder structure, as well as attention mechanisms to generate context-aware predictions for skip behavior in a given Spotify listening session.

4. Data Preparation & Features

SEQUENTIAL MUSIC SKIP PREDICTION

In this section we present our method. First, we present the Spotify dataset and its features for both the sessions and the tracks. We thereafter present an overview of the model, and then go into greater detail of each component of the model.

4.1 Features

We first describe the dataset and features to establish a common terminology to use throughout the paper, and then the feature engineering used to process the data. The dataset consists of two primary parts consisting of a user log and a track dataset. The user log contains the user sessions, which for each user is a sequence of track playbacks. A track playback contains information about the user in general (e.g., if they are premium or the day of the week they are listening), features related to a single-track playback (e.g., what action the user took to end up listening to the current track, or which action the user took to stop listening to this track), and lastly the id of the track being listened to. The training data contains the playback track features for all tracks in the user session. For the testing data this is available for the first half of the session, while the last half of the session only contains a track id and position in the session of the track being played. The track dataset contains a number of features for each track which both relate to meta information about the song (e.g., popularity and release information) and specific information related to the musical content of the song (e.g., beat strength or flatness).

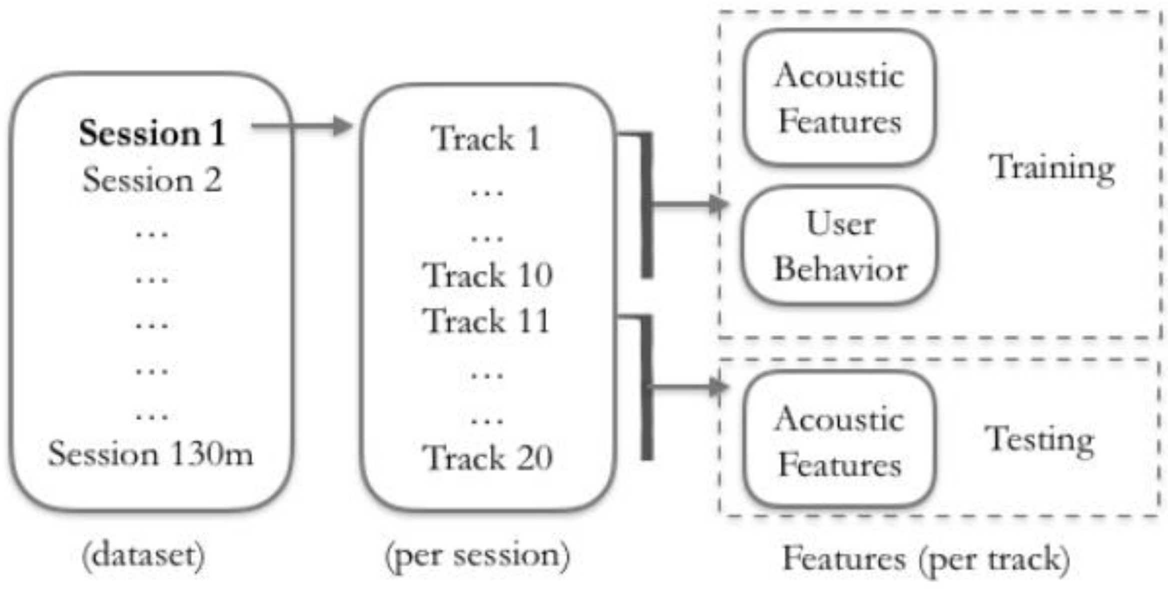


Figure 1: Dataset Structure

4.1.1 Feature processing

Our method is not reliant on any extensive data pre-processing, so the data preparation is straight forward: the user session is represented as 3 types of data:

(1)Meta information associated with the whole session
(2) A sequence of playback tracks for the first half of the listening session
(3) A sequence of track ids and position in the overall session for the second half of the listening session. This was done to mimic how the testing data was constructed.

The meta information (1) for the whole session consists of whether the user is a premium user, the length of the session, and the day of the week. These are encoded separately using a one hot encoding. The first half of the listening session (2) contains all the features for each playback track, except for the features listed in the meta information (1). All categorical features are encoded using a one hot encoding. The second half of the listening session (3) only contains the track id and position in the session for each track.

The track data is standardized such that each feature has 0 mean and unit variance, and is otherwise used as is for representing a track.

4.2 Differing Session Lengths

User sessions are not of a uniform length, being between 10 and 20 tracks long. Considering a session with 20 tracks, given data for x_1 through x_{10} , we want to predict x_{11} through

$$x_0. x_1, x_2, \ldots, x_{10}, x_{11}, \ldots, x_{20}$$

However, for a session with 10 tracks, given data for x_1 through x_5, we want to predict x_6 through x_{10}.

$$x_1, \; x_2, \ldots, x_5, x_6, \ldots, \; x_{10}$$

The typical method of dealing with time series of differing lengths (for example, sentence lengths in natural language processing), is to pad the shorter sequences. However, this is inappropriate for this dataset as we are always interested in the predictions for the second half of the session. Given the differing positions of the tracks needed to be predicted, I re-indexed features and labels of the data so that the tracks in the first half of the session were given by x_{-9} through x_0, while tracks to be predicted were given by x_1 through x_{10}.

4.3 Handling Data

The data were handled in the following manner:

Categorical Features. Categorical features (e.g context type) were transformed using one-hot encoding.

Scaling. Some features associated with the tracks themselves (e.g. release year, duration) required rescaling to the range [0,1].

Missing Data. Due to the treatment of differing session lengths described above, shorter sessions will have missing features or labels. For the purposes of this preliminary investigation, I have replaced missing data with zeros. However, since a zero indicates the absence of a feature or that a track has not been skipped, this may contribute additional noise to the data, and lead a model away from the signal in the data. Further work needs to be done on the best way to treat missing data due to differing sequence lengths.

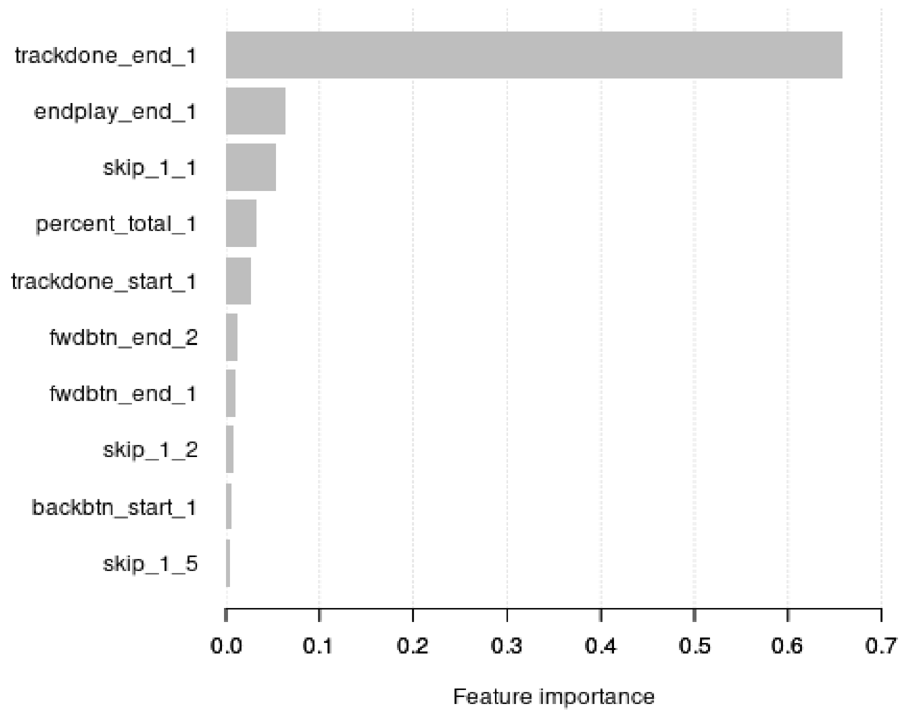
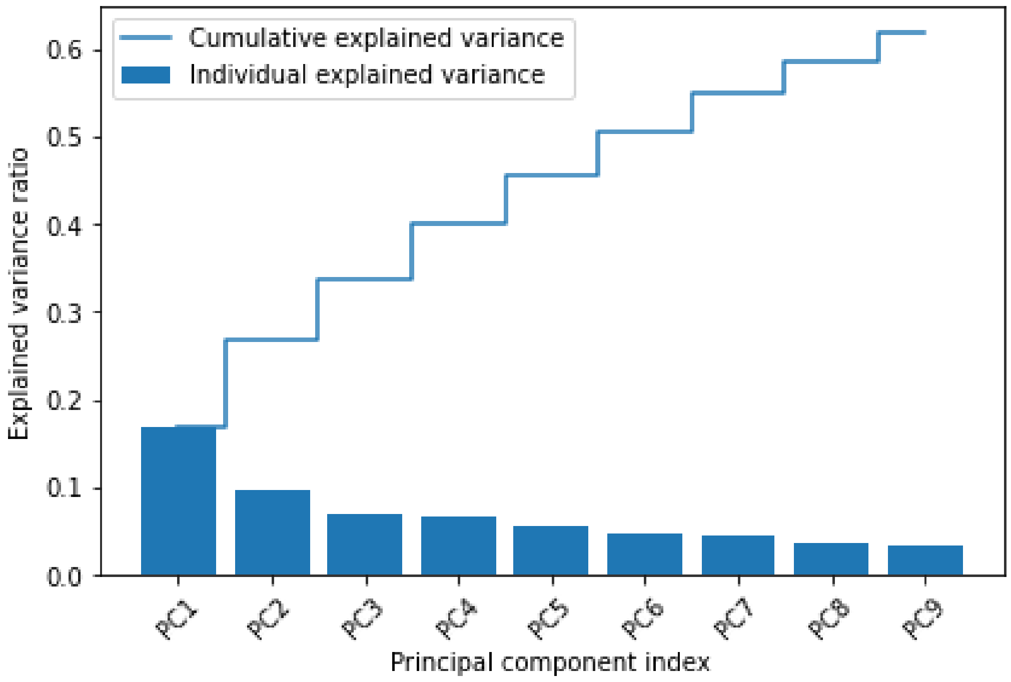


Figure 1 shows the importance of features for predicting the first track in the second half of a session. The most important feature, by a significant margin, is that the final track of the first half of the session was completed. This feature remains the most important for all track positions, but gradually decreases in importance. At the same time, the skip probability of the track in the training data increases in importance.

Figure 1. Feature importance for the first track to be predicted.

4.4 Principal Component Analysis

Principal component analysis is performed on all the track features keeping apart the meta information. The aim of this analysis is to perform Dimensionality reduction and still maintain their importance.



The figure above shows how it takes 9 principal components to represent 60% of the data.

4.5 MODELLING

Two types of models were attempted:

- gradient boosted trees =
- long short-term memory (LSTM) neural networks

The key Python packages used for these models were XGBoost and Keras, respectively. Due to the volume of data available, and the preliminary nature of this investigation, samples were made by randomly selecting a small number of files in the training set to build each model. A selection was also made of files to test the final models against (the hold out set).

4.5.1 Gradient Boosted Trees

A model was trained for each label, or target, track position. That is, a model was trained for the first track position to be predicted, another for the second track position to be predicted, and so forth. Identifying good parameters for training the model involved a grid search.

The parameters ultimately used in training the model were:

- **eta = 0.05**
- **nrounds = 100**

This model achieved a score as:

- **Average precision: 0.556**
- **First prediction accuracy: 0.742**

5. Methods

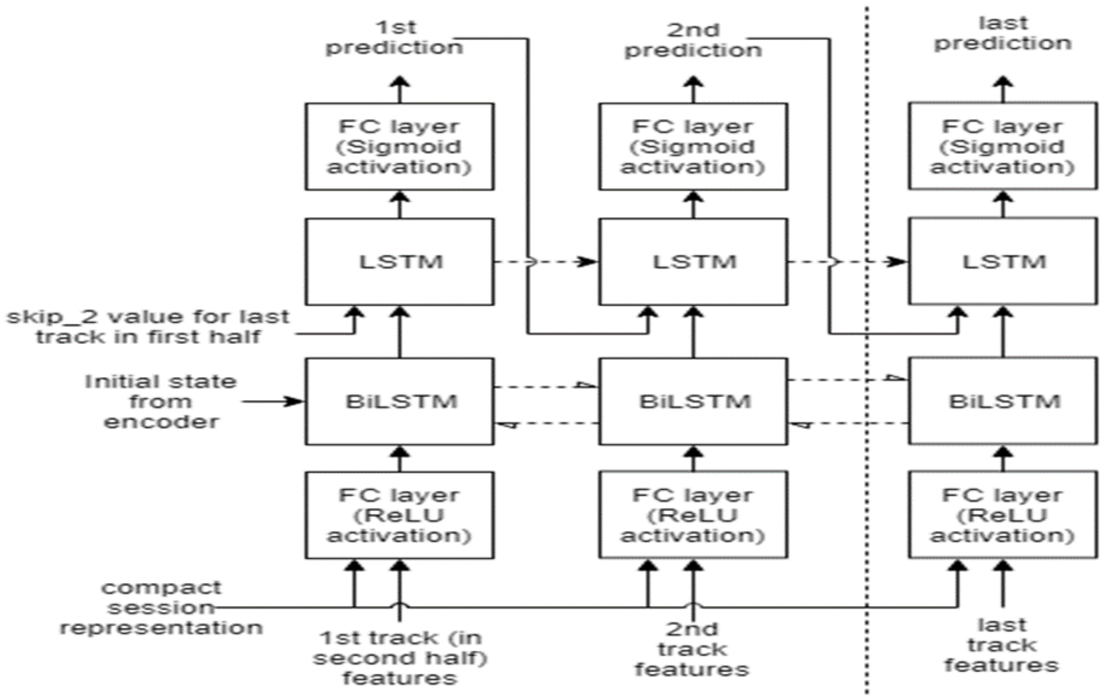
5.1. Preliminary Baseline: Gradient Boosted Trees

We started with using the Gradient Boosted Trees (GBT) algorithm as our baseline because the boosting method in GBT generates predictors sequentially and builds off of previous data (Zhang & Haghani, 2015); i.e., the idea of Gradient Boosted trees is to improve upon the predictions of the first tree in order to build the second tree, and so on. This method of building onto previous data aligns with the task we were solving: sequence-based classification where we want to build a model for each track, we are predicting on based on the previous track. More specifically, for the algorithm itself, we decided to implement Light GBT, due to its high speed, low memory, and capacity to handle large datasets.

5.2. Recurrent Neural Baselines: LSTM and Bi-LSTM

The nature of the task - building a sequential skip prediction model - lent itself towards the use of models that could understand sequential behavior. In this sense, recurrent neural models like LSTMs and Bi-LSTMs are shown to have great capacity to comprehend sequence-based, variable length data when compared to other static deep learning models. This architecture uses two

separate recurrent neural models, the first to encode the inputs into a single highdimensional state, and the second to decode the state passed from the encoder towards the defined prediction task. In our case, we used this encoder-decoder architecture to train and evaluate on our sequence-to-sequence skip prediction task. The track acoustic and user behavior embeddings for the first half of the listening session are passed as input into the encoder model. The resulting hidden state from the encoder is then passed into the decoder model where it is joined by the input features representing the acoustic attributes of each track from the last half of the listening session. This decoder model determines the most likely label for each track. The choice of using LSTMs and Bi-LSTMs as our recurrent neural models was chosen due to the dominance of LSTMs in the world of sequential data over other recurrent neural models, such GRUs, due to their ability to forget, maintain, and regulate cell memory between states.


1st prediction
2nd prediction
last prediction
FC layer (Sigmoid activation)
FC layer (Sigmoid activation)
FC layer (Sigmoid activation)
LSTM
LSTM
LSTM
skip_2 value for last track in first half
Initial state from encoder
BiLSTM
BiLSTM
BiLSTM
FC layer (ReLU activation)
FC layer (ReLU activation)
FC layer (ReLU activation)
compact session representation
1st track (in second half) features
2nd track features
last track features

5.3.Transformer: Traditional NLP Method

Due to the sequential nature of our data, when investigating what advances in other domains may be applicable to our problem and while experimenting with the LSTM/BiLSTMs, we found many parallels to NLP. Like many natural language problems, our challenge required a model that could leverage the sequential structure of the data and understand both long and short-range dependencies. Upon investigating the current advances in NLP, we were drawn to the Transformer and the introduction of attention mechanisms and positional encodings in the encoders and decoders. These Transformers were able to learn long and short-range dependencies in a way that LSTMs and RNNs could not, which is especially important in our dataset because when trying to predict skip/no skip for a certain track, it is important to take into account the audio features for all the previous tracks as well as historical skip behavior. Thus, we decided to develop the architecture for the traditional transformer model used in NLP tasks:

In order to develop this model, we started with creating a meaningful embedding of the user behavior and acoustic features for each track in a session (i.e., 20 tracks). Since we are predicting on the last 10 tracks of a session, we padded and passed in only the acoustic features for these tracks into the embedding. Then, the embedding gets passed into the encoder, creating a hidden state, which gets passed into the decoder along with the output target prediction sequence for the last 10 tracks. The output target sequence that gets passed into the decoder is

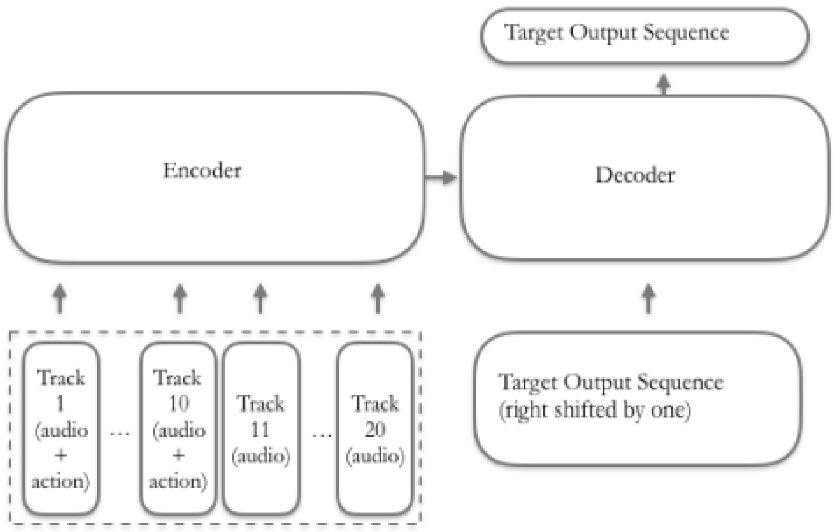


Figure 3: Our Simplified Diagram of Traditional Transformer

right-shifted by one so that the decoder uses the output prediction of the previous track and properly predicts the skip behavior of tracks in the correct order. We finally added a linear layer at the end of our transformer to map our predictions into our labels space (skip/no skip).

5.4. Transformer: Feature-Forcing Method

We also decided to take another novel approach to solving this challenge: enhancing the architecture of the traditional NLP transformer model for our specific task and data. Thus, the question became" how might we leverage the structure of the transformers to predict skips?". Transformers do well with taking an input sequence and converting it into an output sequence (i.e., language translation). However, our problem is unique in that only the second half of the input sequence (session) needs predictions, while the first half has ground truth labels, we want to leverage in the second half's predictions.

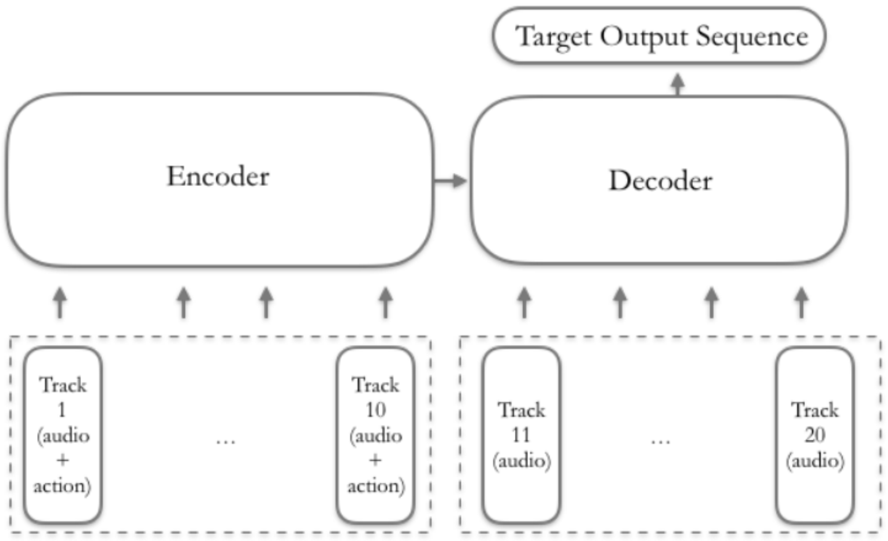


Figure 4: Architecture of our Feature-Forcing Transformer

6. Experiments and Discussion

6.1. Accuracy Metric

The evaluation metric used in the original Spotify Sequential Skip Prediction Challenge is mean Average Accuracy (mAA or MAA). For a single example session i with T tracks, we can define Average Accuracy as AA = 1 T X T i=1 A(i) · 1{y (i) = ˆy (i)}

where y (i) is the ground truth label for track i, yˆ (i) is the predicted label for track i, and A(i) is the accuracy of label predictions in the sequence up to and including the ith track. It should be noted that the expected result of Average Accuracy (AA)is much different than that of conventional accuracy. This is because AA acts like a weighted sum of accuracy that places emphasis on correct classifications earlier in the sequence. Unlike conventional accuracy, AA cannot achieve a value above 0.6 even if just the first label is misclassified.

6.2. Loss Functions

For our experiments, we leveraged two kinds of loss functions to train our models. The first loss function explored was binary cross-entropy loss. For a single session with T tracks, we can define cross-entropy loss as

CE Loss = − 1 T X N i=1 y (i) log p(y (i))+(1−y (i)) log(1−p(y (i)))

where y (i) is the ground truth label for track i and p(y (i)) is the predicted probability of track i being labeled positive. Cross-entropy loss, while useful for maximizing accuracy generally, does not adequately maximize Average Accuracy. As a result, we defined an experimental loss function that combined aspects of minimizing cross-entropy loss while also maximizing Average Accuracy. For a single session with T tracks, this new loss function can be defined as

CEAA loss = − 1 T X T i=1 y (i) log p(y (i)) + (1 − y (i)) log(1 − y (i)) + A(i) · 1{y (i) = ˆy (i) }

where y (i) is the ground truth label for track i, yˆ (i) is the predicted label for track i, and A(i) is the accuracy of label predictions in the session up to and including track i (We also refer to this as Cross Entropy + MAA in our paper).

6.3. Discussion and Error Analysis

Starting with the results of our baseline, we see that the Gradient Boosted Trees (GBT) model achieves 0.51620 subset accuracy on our data. It is important to note that the way we passed input into this model is different compared to our other models, since GBT was meant to be an initial, simple test. Thus, we believe GBT may be achieving around 50% accuracy due to random guessing about whether a track is skipped or not skipped.

Transformers (Traditional NLP):

Based on the output, we see that this transformer did not help improve accuracy, dropping to an accuracy lower than the LSTM. Mapping from an input size of 20 tracks to an output size of 10 tracks may not be the best representation of our problem with sequence-based classification for the last 10 tracks. The separation between the first and second half of a session is less pronounced in this model as all tracks are sent together into the encoder.

Transformer (Feature-Forcing):

In this model, the encoder represented meaningful information about the first half of a session and the decoder paid attention to both the representation of the first half and the individual audio features for each track that needs to be predicted. We see from the results that this architecture proved to be a better model for the problem at hand. We also see that this transformer does not overfit and achieves a higher accuracy than the traditional NLP transformer, LSTM, and Bi-LSTM models.

6.4. Feature Analysis:

LSTMs We can also learn about the relative importance of each input feature by examining a heatmap of their weights on a simple logistic regression model. In particular, we trained a logistic regression model to predict the skip label of the 11th track in a session where the input is a concatenated vector of the 67 acoustic and user behavior features for the first 10 tracks and the 29 acoustic features of the 11th track. Figure 5: Heatmap of relative weight importance of each feature across increasing L1 regularization penalty by analyzing the above heatmap, we can immediately notice that as L1 regularization penalty increases, the activations of the features related to the first 9 tracks (feature indices 0 to 602) greatly diminish. Features from the 10th track, however, are maintained across regularization penalties. We can also note, however, that the features relating to the acoustics of the target track (indices 670 to 699) also zeroout as regularization increases, indicating that even a track's acoustic attributes are not entirely important to its skip label prediction. In fact, by the regularization penalty of 1 and further, what we see is that the most important features relate to the user's behavior on the previous track. Among these features are attributes all heavily related to track playing behavior i.e., if the user ended the 10th track by pressing forward or letting it finish, if the user briefly played the 10th track before skipping, and if the user skipped the 10th track altogether. This indicates that the most important traits towards predicting the skip label of any given track is the user's behavior for the immediately preceding track.

7. Future Work

Given more time, we want to explore the transformer architecture more. Based on the transformers implemented successfully in NLP applications and the performance of our feature-forcing transformer, we could combine aspects of the traditional NLP transformer with the feature-forcing transformer in order to create a better model. We could dynamically append the output prediction for each track from the decoder with the track's audio features and injected these concatenated embeddings into our decoder for prediction. This architecture would perhaps better follow the transformers that are known to work well in NLP, as well as would better represent our model by considering the audio features that are inputting into the decoder. Finally, exploring variable length sessions and improved feature analysis would help us better understand and improve our model

Team Member's

Prateek Smith Patra (Team Lead)

Alka Roy (Team Lead)

Snegha Raghu

Nilesh Misra

Durgara Opisini | Arbaaz Ali Jafri | Shreeja Mishra