Thank you, Next: Using NLP Techniques to Predict Song Skips on Spotify based on Sequential User and Acoustic Data

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.

1. Introduction

As streaming services like Spotify become the go-to providers for music, it is increasingly important that these platforms are able to recommend the right music to their users. Machine learning has long been leveraged to build recommender systems in order to recommend music. This can be seen through improvements made to music streaming platforms, such as customized playlists and suggesting a user new releases that they might like. However, there has not been much research done on how a user interacts with music sequentially, over the course of one listening session. Skipping behavior serves as a powerful signal about what the user does and doesn't like. For instance, in the afternoon, a user might be looking for classical study music, and thus skipping hip-hop. Later that evening, the user might skip classical for hip-hop. Being able to use skip behavior in the context of an entire listening session is key to recommending relevant content, and yet there has been a lack of research into this specific recommendation task.

The focus of our project is to build a sequential skip prediction model to predict if a user will skip over a track based on the user's interactions with previous songs and the song's musical qualities in an individual music listening session. The input will be be an embedding of audio features and user behavior and the output will be predicting whether a track is 'skipped' or 'not skipped'. 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.

2. Related Works

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. (Béres 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.

3. Dataset and Features

We are using the Spotify Sequential Skip Prediction dataset for our project (Brost et al., 2019), which consists of roughly 130 million listening sessions. Each session has at most 20 music tracks. All the user behavior features and track ids are provided for the first half of the session, but only the track ids are provided for the second half. The track ids are associated with the track's acoustic features that can be extracted via the Spotify API. The user behavior features consist of actions like whether the user paused, hour of day, etc. and acoustic features consist of data like danceability, tempo, etc.

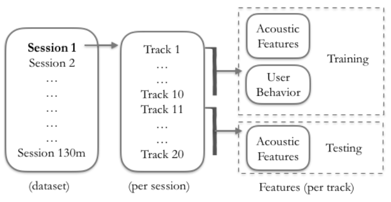


Figure 1: Dataset Structure

In order to pre-process the data for training, we merged user behavior and acoustic features using track ids for each track in the first half of a listening session. We then pre-processed categorical features into one-hot representations and normalized them (using either z-score and min/max). This process created an input track embedding for our model. For the tracks used in the testing phase (tracks from the second half of the session), we only used the pre-processed acoustic features embeddings.

We chose to use the 'skip 2' feature as our output label for whether a track was skipped/not skipped. This is because 'skip 2' indicates whether a user skipped a track early on (i.e. a third of the way through a track), whereas other the skip features 'skip 1' and 'skip 4' respectively indicate whether a user skipped the track almost as soon as it played or near the end of the track. 'Skip 2' is more ideal in indicating the user's actions of listening to a song for a little bit and figuring out if they like/dislike that song (since many people tend to skip songs they like even halfway through).

4. Methods

4.1. 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.

4.2. 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 high-dimensional 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.

4.3. 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/Bi-LSTMs, 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 (Figure 2) and apply it to our task.

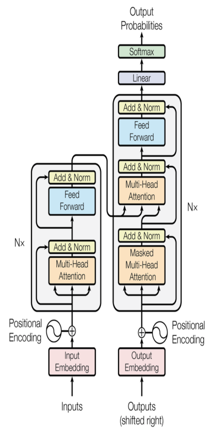


Figure 2: Architecture of Traditional NLP Transformer

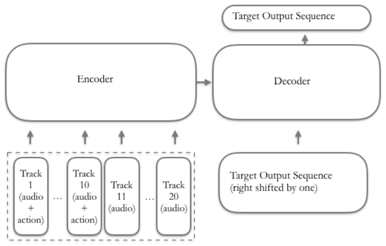


Figure 3: Our Simplified Diagram of Traditional Transformer

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 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).

4.4. 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. With this problem structure in mind, we came to create our own model: The Feature-Forcing Transformer (Figure 4).

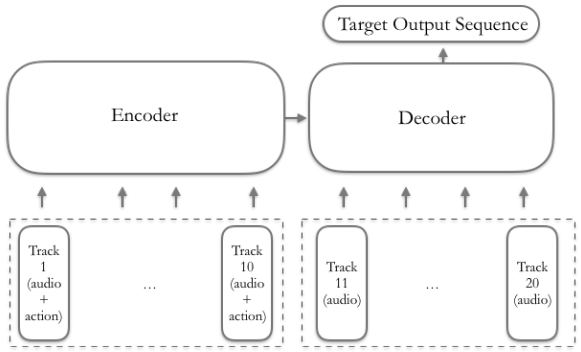


Figure 4: Architecture of our Feature-Forcing Transformer

In the Transformer, the encoder outputs a set of attention vectors K and V, which the decoder utilizes in the encoder-decoder attention layer. This layer allows the decoder to learn which parts of the input sequence it should pay attention to. For our feature-forcing model, we fed only the first 10 tracks (with behavior and audio features) into the encoder, hypothesizing that the encoder would produce an output that meaningfully represents the relevant information from the first 10 tracks.

Then, typically for a Transformer, the decoder takes in a starting token as the first input for target output 0, and then for the following k decoding steps, the k-1 target output of the decoder is taken in as an input. One alternative method that is applied to training the decoder is called ”Teacher forcing”. Teacher forcing helps the model by passing in the true labels of a sequence into the decoder as opposed to looping back in and sending in the labels generated by the decoder. Our ”Feature-forcing” method similarly disregards the decoder output and sends in predetermined features instead. For a target track i, instead of sending in the previous prediction of ’skip 2’ for a target track (i-1), we send in the audio features of the track i. Thus, in the decoder, the model combines the audio features of track i along with the output of the encoder (which encodes audio and behavior for the first half of the session).

5. Experiments and Discussion

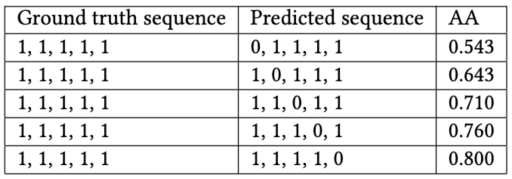
5.1. Accuracy Metric

The evaluation metric used in the original Spotify Sequential Skip Prediction Challenge is mean Average Accuracy (m-AA or MAA). For a single example session i with T tracks, we can define Average Accuracy as

$$AA = \frac{1}{T}\sum_{i=1}^{T} A(i) \cdot 1\{y^{(i)} = \hat{y}^{(i)}\}$$

where $y^{(i)}$ is the ground truth label for track i, $\hat{y}^{(i)}$ 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. The table below demonstrates this.



Ground truth sequence	Predicted sequence	AA
1, 1, 1, 1, 1	0, 1, 1, 1, 1	0.543
1, 1, 1, 1, 1	1, 0, 1, 1, 1	0.643
1, 1, 1, 1, 1	1, 1, 0, 1, 1	0.710
1, 1, 1, 1, 1	1, 1, 1, 0, 1	0.760
1, 1, 1, 1, 1	1, 1, 1, 1, 0	0.800

Table 1: Demonstrates how Average Accuracy can vary greatly given different prediction sequences.

5.2. 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

$$\text{CE Loss} = -\frac{1}{T}\sum_{i=1}^{N} y^{(i)} \log p(y^{(i)}) + (1-y^{(i)}) \log(1-p(y^{(i)}))$$

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

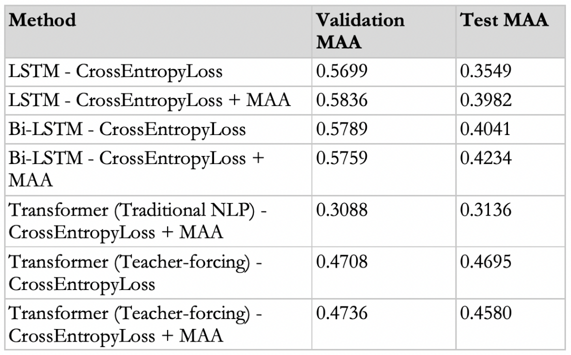
$$\text{CEAA loss} = -\frac{1}{T}\sum_{i=1}^{T} \Big[y^{(i)} \log p(y^{(i)}) + (1-y^{(i)}) \log(1-y^{(i)}) + A(i) \cdot 1\{y^{(i)} = \hat{y}^{(i)}\} \Big]$$

where $y^{(i)}$ is the ground truth label for track i, $\hat{y}^{(i)}$ 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).

5.3. 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.

Below is a datatable with the rest of our models' accuracies.



Method	Validation MAA	Test MAA
LSTM - CrossEntropyLoss	0.5699	0.3549
LSTM - CrossEntropyLoss + MAA	0.5836	0.3982
Bi-LSTM - CrossEntropyLoss	0.5789	0.4041
Bi-LSTM - CrossEntropyLoss + MAA	0.5759	0.4234
Transformer (Traditional NLP) - CrossEntropyLoss + MAA	0.3088	0.3136
Transformer (Teacher-forcing) - CrossEntropyLoss	0.4708	0.4695
Transformer (Teacher-forcing) - CrossEntropyLoss + MAA	0.4736	0.4580

We see that the LSTM and Bi-LSTMs both overfit to the data since the validation MAA is higher than the test MAA. This could be fixed by more rigorous hyperparameter tuning, which we focused on carrying out for our best model: the transformers. Furthermore, we can see that the Bi-LSTM performs slightly better than the LSTM, perhaps due to the more detailed hidden encoder representation of the track sequence in the Bi-LSTM.

Finally, we can make some important observations about the developed transformers:

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.

We have also done some feature analysis on this feature-forcing Transformer. Since the attention mechanism is used to weight specific encoder outputs of the input sequence, we can visualize what the transformer is focusing on at each time step:

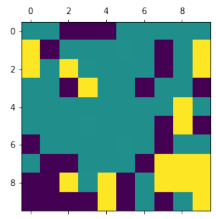


Figure 4: The Encoder-Decoder Attention Weights Visualized

In the image above, we see the attention mechanism as a matrix. The columns correspond to the input steps, and rows correspond to the output steps. Interestingly, we notice that each of the decoder outputs especially focus on the last few tracks from the encoder sequence. One reason this might be is because the skip/no skip behavior of the last track often is a strong indicator of behavior in the future, more so than earlier track skipping behavior.

Furthermore, we explored a variety of different hyperparameters on this successful Transformer, especially the size of the hidden state. After trying 64, 128, and 256, we found that the optimal hidden state was size 128. Perhaps 64 was not a large enough size to fully express the input sequence, and 256 was too large and also wasn't able to convey meaning. We also explored batch size, but after several epochs we found no significant difference in accuracy. For batch size, we settled on 64 per batch to speed up training time without sacrificing accuracy. With more hyperparameter tuning, we likely could have boosted up the MAA.

In the end, this novel reformulation of the Transformer proved to be a moderate success. Our feature-forced Transformer enjoyed a MAA of 0.458, which places us at 29th place in the online submissions for the Spotify Sequential Skip Prediction Challenge.

5.4. 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.

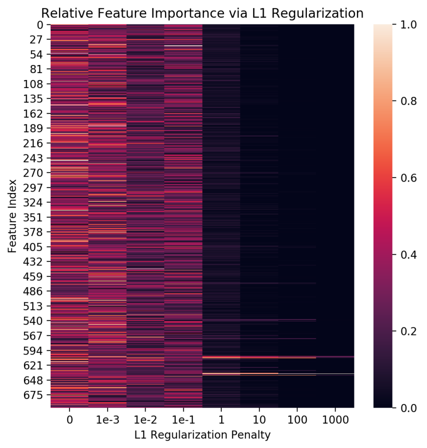


Figure 5: Heatmap of relative weight importance of each feature across increasing L1 regularization penalty

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 zero-out 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.

6. 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.