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

Alex Hurtado (hurtado@stanford.edu), Markie Wagner (markie@stanford.edu), Surabhi Mundada (surabhim@stanford.edu)

Introduction

Music consumption habits have changed dramatically with the rise of streaming services like Spotify, Apple Music, and Tidal. The skip button plays a large role in the user’s experience, as they are free to abandon songs as they choose. Music providers are also incentivized to recommend songs to their users in order to increase user experience and time spent on the platform. Machine learning in the context of music often uses recommender systems. There hasn’t been much research into how a user’s interaction with music over time can help recommend music to the user.

Data and Preprocessing

Dataset: Spotify Sequential Skip Prediction dataset [2], consisting of roughly 130 million listening sessions with associated user behaviors. Each session consists of multiple music tracks (songs, podcasts, etc.). User interaction features are provided for the first half of the session, but only track ids are provided for the second half.

Pre-Processing: We merged user behavior and acoustic features for each track using the track ids. We also preprocessed categorical features into one-hot representations and normalized them (z-score and min/max). This pre-processing created an input track embedding for our model.

Features Details: We chose to use “skip_2” as our output label for whether the song was skipped/not skipped since it better represents whether a user likes the track they are on.

Accuracy Metric

Our evaluation metric on the task of sequential skip prediction is defined as mean average accuracy (MAA). For a single example session, we can calculate average accuracy as \( \text{MAA} = \frac{1}{|A(i)|} \sum_{i \in A(i)} \left(1 - \frac{|F(i)|}{|F(i)|}ight) \), where \( |F(i)| \) is the number of tracks in the given session, \( |A(i)| \) is the ground truth label for track \( i \), and \( A(i) \) is the accuracy in the session up to track \( i \).

We leveraged two loss functions for our task. The first is a simple binary cross-entropy loss function, and the second is calculated as the binary cross-entropy loss minus average accuracy per batch. For a single example session, we define:

\[
\text{Loss} = -\frac{1}{2} \left( y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \right) + A(i) \left(1 - \frac{|F(i)|}{|F(i)|}\right)
\]

Doing the same procedure for our cell gates, we see that features like session position and session length are highly activated, implying that our LSTM learns to place importance on maintaining memory of a track’s relative position in session! Additionally, an analysis on our output gates reveals that our model learns to pass features representing the user skipping the previous track, the user playing a personalized playlist, and the tracks’ US popularity estimate, dynamic range mean, and fluxness.

Evaluation

As our recurrent neural models, we opted to use LSTMs and BiLSTMs. Specifically, we leveraged LSTMs and BiLSTMs in an encoder-decoder architecture to train and evaluate our sequence-to-sequence skip prediction task. In the world of sequential deep learning models, LSTMs have experienced broader use over GRUs due to its gate architecture and it ability to forget, maintain, and regulate cell memory between states. Compared to LSTMs, bidirectional LSTMs are able to perform better on classification tasks since they are able to capture information from both present and future states. The results from our experiments are given below:

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation MAA</th>
<th>Test MAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.5490</td>
<td>0.5490</td>
</tr>
<tr>
<td>LSTM + Feature-Forging</td>
<td>0.5500</td>
<td>0.5500</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>0.5400</td>
<td>0.5400</td>
</tr>
<tr>
<td>BiLSTM + Feature-Forging</td>
<td>0.5400</td>
<td>0.5400</td>
</tr>
</tbody>
</table>

Feature Learning

We can learn about the relative importance of each input feature by examining a heatmap of their activations on the encoder of our LSTM model across each track in the entire listening session. The embedding gates pass into our encoder, creating a hidden state, which gets passed into the decoder along with the target output sequence. 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 in-sequence. We added a linear layer at the end of our transformer to map our predictions into our labels.

Feature Analysis and Explainability for the Transformer: This will help us better understand how the attention layers work in the model and which features are more significant for prediction.

Novel NLP Techniques for Sequential Classification

Transformer - Traditional NLP Model

Inspired by the success of transformers in tackling a variety of different NLP tasks, we were interested in leveraging a transformer, attention, and positional encodings for a sequential classification task, such as the one we are solving.

We developed a meaningful embedding of the user behavior and acoustic features for each track in the entire listening session. The embedding gates pass into our encoder, creating a hidden state, which gets passed into the decoder along with the target output sequence. 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 in-sequence. We added a linear layer at the end of our transformer to map our predictions into our labels.

Table 2: Subset Accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation MAA</th>
<th>Test MAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (Traditional)</td>
<td>0.3807±0.04</td>
<td>0.3333±0.04</td>
</tr>
<tr>
<td>Transformer (Feature-Forging)</td>
<td>0.3807±0.04</td>
<td>0.3333±0.04</td>
</tr>
</tbody>
</table>

LSTM/Bi-LSTM: We can see that both of these models are outperforming the traditional NLP transformer. The BiLSTM performs slightly better than LSTM.

Transformers (Traditional NLP): Based on the output, we can see that our transformer which we developed did not necessarily help improve accuracy. Mapping from an input size of 20 and output size of 10 (tracks) may not be the best representation of the problem. While the sequential nature of data is modeled, the separation between the first half of the session and the second half of the session is less pronounced in this model, as all tracks are sent together.

Discussion/Error Analysis

Gradient Boosted Trees achieves around 50% accuracy, more than the sequential based models, perhaps because of random guessing and a large difference in this model’s architecture and input processing as compared to our other models.

Conclusion

Based on the output, we can see that the transformer which we developed did not necessarily help improve accuracy. Mapping from an input size of 20 and output size of 10 (tracks) may not be the best representation of the problem. While the sequential nature of data is modeled, the separation between the first half of the session and the second half of the session is less pronounced in this model, as all tracks are sent together.

Future

Given more time, we would love to explore some of the following topics:

- Improve Transformer model: Dynamically append the output prediction for each track from the decoder with the audio features and inject an embedding of this into our decoder
- Have our models work with variable length tracks
- Feature analysis and explainability for the Transformer: This will help us better understand how the attention layers work in the model and which features are more significant for prediction

References


Acknowledgements

We would like to thank the CS220 course staff for all their help on this project. We would also like to thank Professor Moses Charikar, Andrew Ng, and Christopher B for providing the opportunity to take this class.