Preliminary Investigation of Spotify Sequential Skip Prediction Challenge

Charles Tremlett
www.cddt.nz
Auckland, New Zealand

ABSTRACT

Given information about the first half of a Spotify user's listening session, the task is to predict whether each track in the second half of the session will be skipped. I outline my initial approaches taken during the challenge hosted by CrowdAI using gradient boosted trees and long short term memory neural networks, as well as highlight the importance of structuring the user session and track data as correct time series. Given the time limitations of the challenge, I conclude with some suggestions for further research which could be done on this dataset.

This solution placed 9th during the challenge. The user name on the CrowdAI platform is "cddt", and the source code associated with this solution can be found at https://github.com/cddt/wsdm-2019-spotify.

KEYWORDS

time series, neural networks, lstm, action prediction, streaming data, user interaction

ACM Reference Format:
Charles Tremlett. 2019. Preliminary Investigation of Spotify Sequential Skip Prediction Challenge. In *Proceedings of WSDM '19: 12th ACM International Conference on Web Search and Data Mining (WSDM '19).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Between November 2018 and January 2019, Spotify, WSDM, and CrowdAI hosted a challenge to encourage research into how users interact with streamed content.

Approximately 130 million listening sessions were provided for the training set, and each session contained between 10 and 20 tracks. [2] The test set comprised approximately 30 million listening sessions, and detailed data was provided for the first half of each session, with the second half of each session to be predicted. Additionally, information about individual tracks were provided for each track which appeared at least once in the training or test sets. In total, the uncompressed data comprised 426 GB, which presented some challenges for standard in-memory processing techniques.

WSDM '19, February 11–15, 2019, Melbourne, Australia

2 EVALUATION METRIC

The primary evaluation metric for this challenge was mean Average Accuracy, given by

$$AA = \frac{\sum_{i=1}^{T} A(i)L(i)}{T}$$

where T is the number to tracks in the second half of the session (i.e. the number of tracks to be predicted), $A(i)$ is the accuracy at the i-th position, and $L(i)$ is a boolean indicator whether the i-th prediction was correct.

The consequence of this metric, is that the first track to be predicted in the sequence has a much greater weighting than subsequent tracks, and the weighting of each track gradually decreases further along in the sequence.

The accuracy of the first track is used as a tie-breaking, or secondary, metric.

As this paper details preliminary investigation, I have not implemented a callback metric for the average mean accuracy as described above. This means that during training, models have been evaluated based on the raw accuracy of all predictions. This is one area where future work may provide some improvement. In this paper, I will be specify the accuracy metric I refer to where necessary (average mean, or raw).

3 DATA STRUCTURE AND PREPARATION

The `R` [8] package `data.table` [5] was invaluable in restructuring the data into a suitable format for modelling.

3.1 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_{20}.

$$x_1, x_2, ..., x_{10}, x_{11}, ..., x_{20}$$

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

$$x_1, x_2, ..., x_5, x_6, ..., x_{10}$$

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}.

3.2 Categorical Features

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

3.3 Scaling

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

3.4 Feature Engineering

The only feature to be manually engineered was the percentage of times the track was skipped in the training dataset. However, some tracks in the test dataset were not present in the training dataset, while others appeared rarely (e.g. once).

3.5 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.

4 MODELLING

Two types of models were attempted:

- gradient boosted trees [6] [3]
- long short term memory (LSTM) neural networks [7]

The key `R` packages used for these models were `xgboost` [4] and `keras` [1], 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).

4.1 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 and 5-fold cross-validation. The parameters ultimately used in training the model were:

- max.depth = 11
- eta = .05
- nrounds = 50
- objective = 'binary:logistic'
- tree_method = 'exact'

Four training files were used for this model. This model achieved a leaderboard score of **0.578** (using a single file achieved a leaderboard score of **0.574**).

This procedure was repeated five times, with different files from the training dataset randomly selected. By averaging the raw predictions, the leaderboard score improved to **0.580**.

The training error for each position to be predicted is shown in **Figure 1**. The error for the first predicted track is substantially lower than for the tracks later in the session. It might be expected that the error continues to increase as tracks later in the session are predicted, and largely that is what is observed. The exception is positions 6 - 10, where the training error is lower than expected, and it is expected that this is due to the treatment of shorter sessions as detailed in section 3.5.

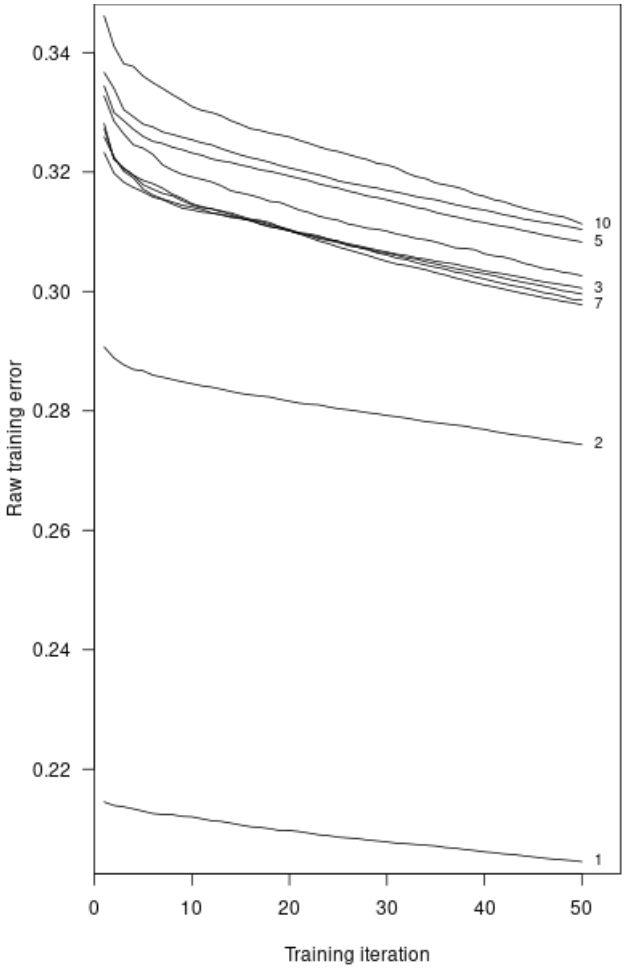

Figure 1: Training error for each track position in the second half of the listening session to be predicted.

Figure 2 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.

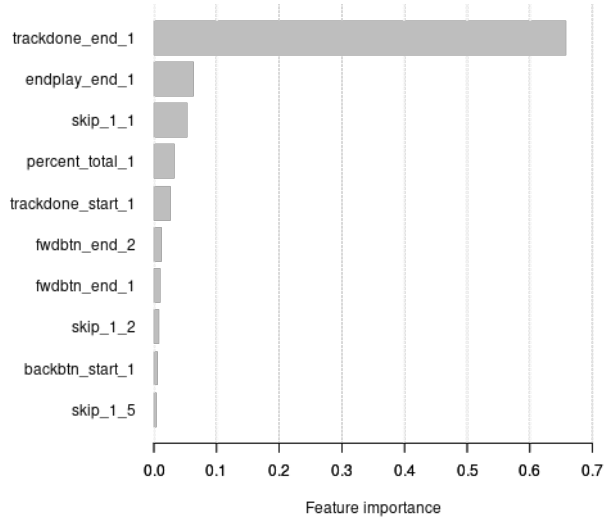

Figure 2: Feature importance for the first track to be predicted.

4.2 LSTM Neural Networks

A recurrent neural network (RNN) was used to model the sequence of skip predictions. Unlike the boosted tree models, a single model was used to predict all ten of the session positions (i.e. multi-label).

The data was prepared into an array with the dimensions (n, t, m), where n is the number of listening sessions, t is the number of time steps in the first half of the session (10), and m is the number of features to be used.

Two bidirectional LSTM layers with m units were found to perform slightly better than a single bidirectional layer. Dropout of 0.2, and recurrent dropout of 0.2, were applied at each layer. The parameters ultimately used in training the model were:

- batch_size = 512
- epochs = 10
- optimizer = 'nadam'
- loss = 'binary_crossentropy'

Furthermore, a smaller sample of data was able to provide the same level of prediction as the boosted tree models. Using a single file in the training set to train this model was able to score **0.582** on the leaderboard.

Repeating the above procedure three times and averaging the raw predictions gave a leaderboard score of **0.583**.

4.3 Ensembling

By averaging the final raw predictions of the gradient tree models and the LSTM models, the leaderboard score improved slightly to **0.584**. While these models were trained using different techniques, the lack of significant improvement indicates that they have likely learned similar signals in the training data.

5 CONCLUSION

5.1 Summary

The models described in section 4 have been shown to be a significant improvement over the best baseline previously available for this dataset (**0.584** vs 0.537).

The two most important features which inform these models are whether the final track of the first half of the session was completed, and the proportion of times the track was skipped in the training data.

5.2 Proposals for further research

The models trained in this paper were primarily designed with time series analysis in mind. A more in-depth consideration of recommendation systems, and how a recommendation system acts in response to dynamic changes to the data (e.g. user actions) is warranted.

Stacking models, especially the boosted tree models, in such a way that the prediction for each track informs the model for the next track, may contribute to some improvement.

Limited use was made of the musical characteristics of the tracks provided. Further work should be done on analysing whether the differences between musical characteristics in subsequent tracks influences the likelihood of a user skipping a track. It is expected that deeper, more complex models, may be required to tease out these influences.

Several opportunities for data augmentation exist, none have been explored here.

Further work could be done on processing larger samples of the training data at once, to determine whether the decision boundary in the data could be better learned. Only a small proportion of the training data was used to train the models described here.

REFERENCES

[1] JJ Allaire and Francois Chollet. 2018. *keras: R Interface to 'Keras'*. https://keras.rstudio.com R package version 2.2.4.
[2] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *Proceedings of the 2019 Web Conference*. ACM.
[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785
[4] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, and Yutian Li. 2018. *xgboost: Extreme Gradient Boosting*. https://CRAN.R-project.org/package=xgboost R package version 0.71.2.
[5] Matt Dowle and Arun Srinivasan. 2018. *data.table: Extension of 'data.frame'*. https://CRAN.R-project.org/package=data.table R package version 1.11.8.
[6] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist.* 29, 5 (10 2001), 1189–1232. https://doi.org/10.1214/aos/1013203451
[7] Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735
[8] R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/