Sequential Skip Prediction using Similarity Measures and Recurrent Neural Networks

Jens Leysen
University of Antwerp
Department of Computer Science
Jens.Leysen@student.uantwerpen.be

***Abstract*—Spotify constantly seeks to improve its user experience. The 2019 Spotify Skip Prediction Challenge was released exactly for this purpose. By predicting whether users will skip certain songs, Spotify can better understand its users' needs and based on these insights improve its music recommendation system. This paper researches the use of similarity measures as a way of improving the skip prediction accuracy of neural network models. Specifically, this study considers the cosine similarity measure calculated on different sets of track features. The usefulness of this measure is evaluated based on a recurrent neural network and gradient boosted trees model. The results were obtained by training these models on a large dataset which consists of an entire day's worth of listening sessions. It was found that the cosine similarity measure doesn't improve the accuracy of either model. This is somewhat unexpected since it has been shown by other teams that a gradient boosted trees' accuracy improves when using other similarity measures such as the Mahalanobis distance.**

***Keywords*—Music Recommendation, Similarity Measures, Skip Prediction, Recurrent Neural Networks**

1. Introduction

In 2019, the Spotify Skip Prediction Challenge was released to inspire further research into the domain of user interaction modelling. Through this challenge, Spotify seeked to improve its music recommendation engine. From an academic standpoint, the Spotify Skip Prediction Challenge offered a unique opportunity for researchers to improve and design state-of-the-art machine learning algorithms using a large, real-world dataset. Although the Spotify Skip Prediction Challenge was concluded in 2019, the challenge description and dataset are still available at: spotify-sequential-skip-prediction-challenge.

The specific goal of the challenge was to predict whether a listener will skip a certain song. After the challenge was concluded in early 2019, the top-performing teams were invited to write papers about how they tackled the challenge. The state-of-the-art uses these three ideas:

- Feature Engineering
- Embeddings
- Recurrent Neural Networks

Embeddings were used by almost all top teams last year. The 'obvious' explanation for this is that the (dis)similarity captured by embeddings is somehow useful to neural networks. This led us to consider the following research question: "What other distance and similarity measures can be used to create useful new features?" In our research, we specifically considered the cosine similarity measure.

This paper continues as follows. In Section 2, the necessary background information is introduced. This section starts with describing the Spotify dataset. In Section 2.2, the relationship between the cosine similarity and embeddings is explained. Afterwards, in Section 2.3 and 2.4, concepts related to feature engineering and recurrent neural networks (RNN) are briefly introduced. In Section 3, we outline the architecture of our RNN model and we discuss the new features that were engineered using the cosine similarity. The influence of these newly created features on the RNN model is evaluated in Section 4. The evaluation makes use of two different performance metrics, mean average accuracy and first prediction accuracy. Each of these metrics is briefly explained at the beginning of Section 4. The paper ends with a conclusion and outline of future work in Section 5 and Section 6.

2. Background

2.1. Dataset

The dataset provided by Spotify consists of two parts, session logs and tracks. The session logs contain 130 million listening sessions [1]. Each of these listening sessions is described by 21 features, such as:

- unique session ID
- sequence of tracks played during the session
- the date and hour the session was played

The tracks dataset contains the four million unique tracks users listened to during their listening sessions [1]. Each of these tracks is described by 29 features, such as:

- unique track ID
- track duration
- track popularity rating
- track beat strength

A graphical overview of the session structure can be seen in Figure 1. In this figure, three sessions are shown. Each session consists of a sequence of tracks where each track is identified by a unique track ID. Skipped tracks are identified by a red box, while tracks that weren't skipped by a user are identified by a green box. The goal of the challenge is to predict the skipping behaviour for the second half of each listening session.

Figure 1: Spotify Dataset: Session Structure

2.2. Vector Embeddings

Vector embeddings gained wide popularity in 2013 when a team of Google researches released their Word2Vec algorithm in this landmark paper [2]. This algorithm has since become central to natural language processing. Vector embeddings effectively reduce a high cardinality categorical variable (such as the words in a language) to a lower-dimensional representation. This new representation puts categorical variables that have similar meanings, connotations or other relationships closer to each other in the vector space. Recently, researchers [3] have started applying vector embeddings to use cases outside of natural language processing. Some of the top teams [4][5][6] explicitly mentioned using a vector embedding or representational learning strategy for the very high cardinality track IDs.

An important feature of embeddings is that they can be pre-trained. Said differently, the process of training an embedding can be completed entirely separately from the main training task. This speeds up the process of training a complex model since the embedding representation doesn't need to be learned at the time of training the main predictive model.

The results from last year's teams show that adding track embeddings to neural networks help to improve the skip prediction accuracy. An intuitive explanation for this is that the (dis)similarity captured by embeddings is somehow useful for the neural networks. In essence, embeddings capture a type of similarity based on how often tracks appear together in a listening session. This is why it's interesting to consider other similarity measures as a possible way of improving the skip prediction accuracy.

2.2.1. Difference with Cosine Similarity. The cosine similarity captures a different type of similarity than track embeddings. With track embeddings, the similarity is indirectly calculated based on the behaviour of users, not the features of a track. Tracks that appear close to each other in a session will be put closer in the embedding's vector space and will thus be seen as more similar. The user creates the playlists and is, as a consequence, responsible for placing together tracks that are similar in some way. Unlike track embeddings, the features we created using the cosine similarity do make use of the track features.

2.3. Feature Engineering

A feature is a numeric representation of raw data, feature engineering is the process of formulating the most appropriate features given the data, the model, and the task [7, p. 15]. A way of improving predictions is to create more meaningful features, i.e. features that help the model to better predict whether a track is going to be skipped or not. Béres et al. [4] defined over 508 features for each track. Some interesting features they created are:

- Track skip information: the skip ratio for every song.
- Session Heterogeneity: the Mahalanobis distance between a song vector and mean representation of the session tracks. But also, how many times a track was repeated in the current session and the unique track ratio.
- Skip-pattern features: for every possible sequence of skip values, calculate the fraction of skip values in the second part of the sessions.

Béres et al. [4] showed that these 3 new feature categories account for nearly 53% of the feature importance in their gradient boosted trees model. Some of these features were also used by Jeunen and Goethals [8], who ended in fifth place. In the context of our research, different features were engineered based on the cosine similarity measure.

2.4. Recurrent Neural Networks

The goal of this paper is not to give an in-depth mathematical introduction to recurrent neural networks. In this section, the history of recurrent neural networks will be explained briefly and further reading related to recurrent neural networks will be presented to the reader. Recurrent neural networks are a class of neural networks that can leverage the information present in sequential data. The top five teams [5][6][8][9][10] all make use of recurrent neural networks, specifically long short-term memory (LSTM) networks and attention mechanisms. However, the specific architectures of the models differ greatly between teams. Over the last few decades, multiple improvements have been proposed over the original LSTM network proposed by Hochreiter and Schmidhuber [11]. For a recent overview, please see Greff et al. [12]. In 2014, a simplified LSTM network named gated recurrent unit (GRU) was introduced by Cho et al. [13]. Both LSTM and GRU are still often used in the

machine learning community. Keras, an open-source library of artificial neural networks, provides implementations of both the original LSTM and GRU networks [14]. Although the model introduced by Cho et al. [13] was state-of-the art in 2014, a year later, Bahdanau et al. [15] introduced the concept of attention mechanisms. Attention mechanisms resolved some issues related to the encoder-decoder model proposed in [13]. This idea of attention mechanisms led to the development of transformers [16] which have been the source of many exciting developments over the past few years. For example, GPT-3 (Generative Pre-trained Transformer 3), a language model that uses transformers to produce text, gained popular attention in the summer of 2020 because of its uncanny ability to produce human-like text [17]. In the context of our research, recurrent neural networks based on LSTM layers were used to evaluate the usefulness of the generated similarity measures.

3. Solution

The code used for this paper can be found here: Djensonsan-Spotify-Sequential-Skip-Prediction-Challenge. The Github project consists of a set of Jupyter Notebooks that implement specific parts of the solution. More information can be found in the README.md file.

3.1. Engineering Similarity Measures

Intuitively, users skip tracks that are dissimilar to their usual listening pattern. By using a similarity measure, we seek to create a feature that captures this dissimilarity between a user's normal listening pattern and a song. Both the Euclidean distance or Cosine Similarity could be used for this purpose. Each one performs better on different types of tasks and which one to choose is often a trial-and-error process. In this paper, we only considered the cosine similarity.

3.1.1. Cosine Similarity. The cosine similarity is given by following formula:

$$cos(\phi) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} \quad (1)$$

Where **A** and **B** represent track vectors. These track vectors each consist of a set of track features, some of which were listed in Section 2.1. For our purpose, we will calculate the cosine similarity between the mean of the session and each individual song. Said differently, **A** will be a vector representing the average of all tracks in a user's session. **B** represents a specific song in the second half of the session. In addition to using the entire set of track features to describe each vector, we will consider following subsets:

- Acoustic features: represented by the features ranging from 'acoustic_vector_0' to 'acoustic_vector_7'.
- Musical features: represented by the following features: 'acousticness', 'beat_strength', 'bounciness', 'danceability', 'dyn_range_mean', 'energy', 'flatness', 'instrumentalness', 'key', 'liveness', 'loudness', 'mechanism', 'organism', 'speechiness', 'tempo', 'time_signature' and 'valence'.

After calculating these cosine similarities, the results are added to the original dataset. Next, this dataset is pre-processed and used to train an RNN and gradient boosted trees (GBT) model.

3.2. Data Pre-processing

Before inputting data into a model, the data needs to be pre-processed. The following processing steps were applied:

- The 'date' column was dropped from the session logs.
- The three categorical columns 'context_type', 'hist_user_behavior_reason_start' and 'hist_user_behavior_reason_end' were one-hot encoded.
- Boolean values were binarized.
- All other features were normalized.
- The session logs were joined with the track information.

These pre-processing steps were implemented as a data pipeline.

3.3. Model Architecture

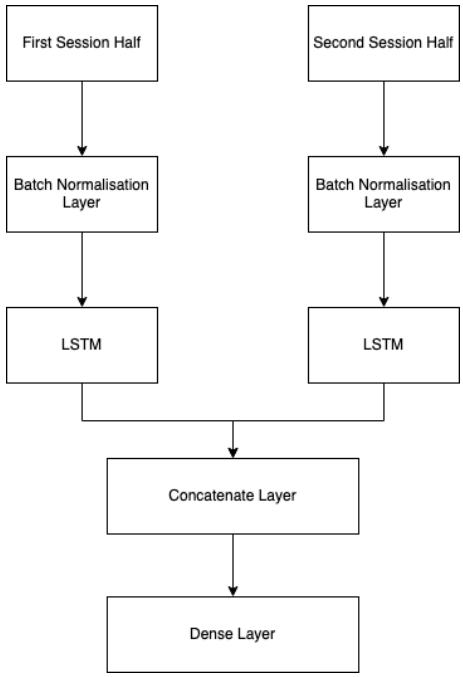

Figure 2: RNN Model

3.3.1. RNN Model. The RNN model makes use of two bidirectional LSTM layers, preceded by two batch normalisation layers. These two layers are joined together and fed

into a final Dense layer with a rectifier activation function. This final layer will produce the prediction. A graphical representation can be seen in Figure 2. The two input layers represent the first and second half of each session. These two layers have different dimensions. The first session half contains both user behaviour and track information while the second session half only includes track information. The exact dimensions of these layers depend on any new features that have been created and added to the dataset.

3.3.2. GBT Model. In addition to the RNN model, a simple GBT model was used to evaluate the engineered features. The specific model uses sklearn's GradientBoostingClassifier class, more information can be found inside the notebooks on Github.

4. Evaluation

All models were trained and evaluated on the same amount of data unless specified otherwise. The training data is $\frac{7}{8}$ of an entire day's worth of listening sessions. The test set is the remaining $\frac{1}{8}$ of that day's listening sessions. This entire dataset contains:

- 23,854,397 rows of session logs
- 1,423,253 unique listening sessions

The results were evaluated based on two performance metrics: the mean average accuracy (MAA) and first prediction accuracy (FPA). The MAA is given by the following formula:

$$\sum_{S \in D} \frac{1}{|S|} \sum_{i=0}^{|S|} Acc(S_{0:i} \times Acc(S_i))$$

This accuracy measure can be interpreted as a weighted sum over the individual predictions where the weight of each prediction is smaller the closer it is to the end of the session. This idea is best illustrated by means a small example:

1) Listed in Table 1 is a prediction and truth value. A one indicates a track was skipped and a zero indicates a track wasn't skipped.
2) Based on the prediction and truth values, the accuracy of S_i (Acc(S_i)) for each position within the session can be calculated. This is shown in row three of Table 1.
3) Based on the prediction and truth values, the accuracy for each position given the total accuracy over the previous values (Acc($S_{0:i}$)) can be calculated. As shown in row four of Table 1.
4) Using the results of the previous two points, the MAA can be calculated as shown in the last row of Table 1. The result is an MAA of 0.543.

Referring to Figure 1, the first prediction accuracy is simply the average prediction accuracy of the first song in the second half of each session.

Prediction	[0, 1 ,1, 0, 1, 0, 0]
Truth	[0, 0 ,1, 0, 0, 0, 0]
Acc(S_i)	[1, 0, 1, 1, 0, 1, 1]
Acc($S_{0:i}$)	[1, 1/2, 2/3, 3/4, 3/5, 4/6, 5/7]
MAA	(1+2/3+3/4+4/6+5/7)/7=0.543

TABLE 1: MAA Example

4.1. GBT Model

Adding the cosine similarity to the dataset decreased the MAA but slightly increased the FPA for the cosine similarity measures calculated on all the features and on the musical features. As can be seen in Table 2, the MAA and FPA both decreased for the cosine similarity measure based on the acoustic vectors. It should be noted that the GBT model was trained on a smaller dataset equal to $\frac{1}{8}$ of a day's listening sessions. The test set remained $\frac{1}{8}$ of that day's listening sessions.

	MAA	FPA
Without cosine similarity	0.570	0.765
With cosine similarity on all features	0.568	0.766
With cosine similarity on acoustic vectors	0.567	0.762
With cosine similarity on musical features	0.569	0.767

TABLE 2: GBT Model: Mean Average Accuracy and First Prediction Accuracy

4.2. RNN Model

As can be seen in Table 3, adding a cosine measure calculated from the vectors containing all track features lowers the MAA of the model. Adding a similarity measure based on the acoustic vectors or musical features doesn't improve or decrease the MAA. The FPA remained constant for all scenarios.

	MAA	FPA
Without cosine similarity	0.603	0.780
With cosine similarity on all features	0.594	0.780
With cosine similarity on acoustic vectors	0.603	0.780
With cosine similarity on musical features	0.603	0.780

TABLE 3: RNN Model: Mean Average Accuracy and First Prediction Accuracy

5. Conclusion

In this paper, we researched how similarity measures could be used to create useful new features and improve the accuracy of models based on neural networks. A cosine similarity measure was created based on multiple sets of features. These distance measures were added to the dataset and used for training an RNN and GBT model. It was shown by evaluating these models on a large dataset that the cosine similarity didn't improve the average precision. This result is somewhat unexpected and requires further research. One

would at least expect the GBT model's accuracy to increase, as GBT models cannot create complex interaction features and their accuracy has been shown to improve by the use of embeddings and the Mahalanobis distance [4].

6. Future Work

As next steps, other similarity measures such as the Euclidean distance and Mahalanobis distance can be researched. Specifically, it needs to be researched why the cosine similarity measure doesn't improve the skip prediction accuracy while other similarity measures do.

References

[1] B. Brost, R. Mehrotra, and T. Jehan, "The music streaming sessions dataset," in *Proceedings of the 2019 Web Conference*, ACM, 2019.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12, 2013.

[3] C. Guo and F. Berkhahn, "Entity Embeddings of Categorical Variables," no. 1, pp. 1–9, 2016. [Online]. Available: http://arxiv.org/abs/1604.06737.

[4] F. Béres, "Sequential skip prediction using deep learning and ensembles," pp. 3–6, 2019.

[5] C. Hansen, C. Hansen, S. Alstrup, J. G. Simonsen, and C. Lioma, "Modelling Sequential Music Track Skips using a Multi-RNN Approach," pp. 1–4, 2019. [Online]. Available: http://arxiv.org/abs/1903.08408.

[6] S. Chang, S. Lee, and K. Lee, "Sequential Skip Prediction with Few-shot in Streamed Music Contents," pp. 3–6, 2019. [Online]. Available: http://arxiv.org/abs/1901.08203.

[7] A. Zheng and A. Casari, *Feature engineering for machine learning*, September. O'Reilly Media, Inc., 2018, p. 218, ISBN: 978-1491953242. [Online]. Available: https://perso.limsi.fr/annlor/enseignement/ensiie/Feature_Engineering_for_Machine_Learning.pdf%0Ahttps://www.amazon.com/Feature-Engineering-Machine-Learning-Principles/dp/1491953241.

[8] O. Jeunen, "Predicting Sequential User Behaviour with Session-Based Recurrent Neural Networks," 2019.

[9] A. Author, "Utilizing Attention Mechanism for Sequential Skip Predcition," 2019.

[10] L. Zhu and Y. Chen, "Session-based Sequential Skip Prediction via Recurrent Neural Networks," *Proceedings of the ACM WSDM Cup 2019, Melbourne, Australia, Feb 11, 2019 (WSDM Cup '19)*, 2019.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[12] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017, ISSN: 21622388. DOI: 10.1109/TNNLS.2016.2582924.

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014. DOI: 10.3115/v1/d14-1179.

[14] Keras, *Recurrent Layers*. [Online]. Available: https://keras.io/api/layers/recurrent_layers/.

[15] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017, ISSN: 10495258.

[17] J. Vincent, *OpenAI's latest breakthrough is astonishingly powerful, but still fighting its flaws*, Jul. 2020. [Online]. Available: https://www.theverge.com/21346343/gpt-3-explainer-openai-examples-errors-agi-potential.