Pairwise Learning to Rank for Hit Song Prediction

Maximilian Mayerl[1] Michael Vötter[2] Günther Specht[3] Eva Zangerle[4]

Abstract: Predicting the popularity of songs in advance is of great interest to the music industry, possible applications include assessing the potential of a new song, automated songwriting assistants, or song recommender systems. This task was traditionally solved by using pointwise models focused on single songs, either using classification to categorize songs into classes like *hit* and *non-hit*, or regression to predict popularity metrics like *play count*. In this work, we draw inspiration from research on learning to rank and instead use a pairwise model. Our model takes a pair of songs *A* and *B* and predicts whether song *A* is more popular than song *B*. We present a neural network model that is trained in a pairwise fashion, as well as two data augmentation strategies for improving its performance. We also compare our model to one trained in a traditional pointwise manner. Our experiments show that the pairwise model using our proposed augmentation strategies outperforms the pointwise model.

Keywords: Learning to Rank; Hit Song Prediction; Pairwise Ranking

1 Introduction

Hit song prediction, also called *song popularity prediction*, is a common task in music information retrieval. Its goal is to predict how popular a particular song is going to be, where popularity is usually defined in terms of sales, streaming counts, or a proxy like chart positions. Solving this task is of high interest to members of the music industry. A record label could use it to screen potential new releases, decide whether to release a record or to determine how much to invest in a particular record's promotion. Musicians could use it to get automated feedback while writing songs. Music retailers or streaming sites could employ it to augment their recommender systems.

In general, hit song prediction is often framed as a classification or regression problem. In the classification case, models are trained to divide songs into classes like *popular* and *unpopular*, while in the regression case, models predict popularity measures such as a song's top chart position or listening counts. As their input, hit song prediction models typically use intrinsic features of the song's audio signal, ranging from low-level features like Mel spectrograms to high-level features like danceability. Some models also incorporate song lyrics.

[1] Universität Innsbruck maximilian.mayerl@uibk.ac.at
[2] Universität Innsbruck michael.voetter@uibk.ac.at
[3] Universität Innsbruck guenther.specht@uibk.ac.at
[4] Universität Innsbruck eva.zangerle@uibk.ac.at

doi:10.18420/BTW2023-26

At its core, hit song prediction can also be viewed as a ranking problem—ordering songs by their popularity provides an implicit ranking of songs by popularity. This is also reflected in how song popularity is often communicated in the form of charts. To model hit song prediction for our approach, we borrow from a related field in information retrieval—learning to rank [Li11]. Most existing approaches to hit song prediction have one thing in common: they seek to predict the popularity—either in the form of a class label or a continuous popularity measure—of a *single* song; they are pointwise ranking models. In learning to rank (LTR), pointwise models have historically been superseded by pairwise and listwise models, which show better results than pointwise approaches. We, therefore, propose to tackle hit song prediction using a pairwise approach, solving the following task: Given a pair of songs A and B, predict whether song A is *more popular* than song B.

The core contributions of this paper are: (1) We propose a neural network model that takes the audio features of two songs as its input and outputs a label $y \in \{0, 1\}$, indicating whether the first song is more popular than the second song. (2) We propose two approaches to augment the data used for training this model, and show that both of these approaches improve the performance of the resulting model. (3) We compare the performance of our model against a model that is trained in the traditional, pointwise way—i.e., as a regressor predicting a popularity score for a single song—and show that the pairwise model performs better for ranking songs by popularity than the pointwise model. (4) For our experiments, we construct a dataset based on data from MusicBrainz[5], AcousticBrainz[6], and Last.fm[7], which we make publicly available.

2 Related Work

Given its big potential impact on the music industry, a body of research has investigated possible solutions, with mixed results. Originally, there was doubt whether predicting song popularity based on audio features of a song was even possible. To that end, Pachet and Roy [PR08] performed a classification experiments using audio features, and concluded that "hit song science is not yet a science". Later approaches were more successful, and used either classification or regression models. The machine learning models employed for this include support vector machines [LL18, DL05], boosting classifiers [DL05], shifting perceptrons [Ni11], and more complex neural network architectures [Za19, Ya17, Yu17]. What most of those approaches have in common is that they only consider single songs at a time; their models take individual songs, and seek to predict their popularity. However, song popularity does not exist in isolation. Songs are competing with each other for the attention and time of music listeners.

Therefore, our approach aims to take this competitive context into account by looking at pairs of songs. To the best of our knowledge, there exists only one similar approach. Yu et

[5] https://musicbrainz.org
[6] https://acousticbrainz.org
[7] https://www.last.fm

al. [Yu17] use a pair of CNNs with shared parameters, which they then train using a loss function incorporating a ranking loss. This makes their CNNs predict popularity scores that reflect the relative popularity ranking of the songs. However, different from our approach, their model still produces a scalar popularity score for every song, whereas our approach aims to directly answer whether one song is more popular than another.

3 Dataset

To conduct our experiments, we require a dataset of songs with both audio features as well as popularity information. Such a dataset needs to have the following properties: (1) It has to be large enough to make training of a neural network model with a large number of parameters feasible. (2) The songs have to follow a realistic distribution of popularity, to avoid introducing biases into models trained on the dataset. (3) The dataset needs to feature a popularity measure that can be used equally well for popular and unpopular songs. (4) Ideally, it should be publicly available or consist only of data that is publicly available, to make our research reproducible. To our knowledge, no such dataset is currently publicly available. Therefore, we propose a new dataset, which we also make available publicly for other researchers via Zenodo[8]. Datasets that were previously used for hit song prediction are either too small (with only a few thousand songs), have a biased (unrealistic) distribution of popularity (e.g., are balanced in hits and non-hits), use popularity measures that are not suitable for unpopular songs (e.g., chart positions only being available for popular songs), or are not publicly available. A comparison of our dataset to the ones used in prior research is given in Table 1. For the construction of our dataset, we used three sources of data: MusicBrainz[9] for song metadata, AcousticBrainz[10] for acoustic features, and Last.fm[11] for play counts as popularity measure. As can be seen from Table 1, our dataset is the largest, features a realistic popularity distribution, and uses a popularity measure that equally addresses popular and unpopular songs. In the following, we describe the creation of the dataset.

3.1 Song Metadata

MusicBrainz is an openly available database of music metadata. It provides information about artists, recordings, releases, etc., and gets its data via crowd-sourcing. We use this database as our ground set of songs. Since MusicBrainz contains information about a large number of songs, both popular and unpopular, we reason that a random sample of songs drawn from its database should give us a set of songs that is reasonably representative in terms of the distribution of their popularity. We do note, however, that songs only known to

[8] https://zenodo.org/record/7525833#.Y77d-7XMJaY
[9] https://musicbrainz.org
[10] https://acousticbrainz.org
[11] https://www.last.fm

Used By	Size	PD	PM	PA
Pachet and Roy [PR08]	32,000	no	yes	no
Dhanaraj and Logan [DL05]	1,700	no	no	no
Lee and Lee [LL18]	16,686	no	no	no
Lee and Lee [LL18]	1,264	no	no	no
Ni et al. [Ni11]	5,947	no	no	no
Zangerle et al. [Za19]	11,646	no	no	yes
Yang et al. [Ya17]	~125,000	no	yes	no
Our Dataset	142,963	yes	yes	yes

Tab. 1: A comparison of datasets used in prior research and our dataset. Size is given in number of songs. PD: Is the popularity distribution of songs in the dataset realistic? PM: Is the popularity measure used in the dataset suitable for popular as well as unpopular songs? PA: Is the dataset publicly available?

very few people are unlikely to have an entry in MusicBrainz, which could introduce a slight bias towards more popular music. For our dataset, we used a dump of the MusicBrainz database obtained at the beginning of 2020. From the songs in the database, we drew a random sample of approximately 20%, providing us with 3,407,667 songs.

3.2 Acoustic Features

After obtaining our ground set of songs, the next step was to gather audio data for these songs. For this, we used AcousticBrainz, a project in the same vein as MusicBrainz to collect a set of rich audio features for as many songs as possible. These audio features are obtained by contributors of AcousticBrainz running a client application, which performs feature extraction using the open-source Essentia library [Bo13b, Bo13a] and then uploads the results to the AcousticBrainz platform. We chose to use AcousticBrainz because (1) its data is indexed by MusicBrainz IDs, making it easy to merge the audio data with our set of songs, (2) it provides audio features for a large collection of songs, and (3) Essentia is a well-known feature extractor which is frequently used in music information retrieval.

For our dataset, we used the newest AcousticBrainz data dump, released in January 2015. This limits our dataset to songs that were released before that date. This way, we were able to obtain audio data for 201,047 of the 3,407,667 songs in our ground dataset. Via AcousticBrainz, we added the following audio features to our dataset, which are a subset of the features provided by AcousticBrainz[12] (we mainly retained features describing the whole song, as opposed to individual time slices of a song): 71 high-level audio features (mood and genre descriptors, gender of the singer, etc.), 236 low-level audio features (loudness, dissonance, spectral features like MFCCs, etc.), 57 rhythmic features (beats per minute, onset rate, etc.), and 45 tonal features (the key of the song, the tuning frequency, etc.).

[12] https://essentia.upf.edu/streaming_extractor_music.html#music-descriptors

Property	Count	Mean	Std. Dev.	Min	Max	Median
Number of Songs	142,963	-	-	-	-	-
Number of Artists	25,667	-	-	-	-	-
Number of Features	409	-	-	-	-	-
Songs per Artist	-	5.57	14.25	1.00	588.00	2.00
Play Count	-	100,046.28	464,534.19	0.00	16,668,020.0	4,325.0

Tab. 2: Summary of the properties of our dataset.

3.3 Popularity Measure

The last step for constructing our dataset was adding a suitable popularity measure. Along the lines of Schedl [Sc16], we used the play counts provided by Last.fm for this. Last.fm is a popular music listening platform that can be queried using MusicBrainz IDs, making it easy to crawl the play counts for the songs in our dataset. Using play counts as a popularity measure has the benefit of providing a natural measure for the popularity of all songs, as opposed to only for popular songs like chart position or similar measures. Of the 201,047 songs with audio features, we were able to obtain a play count for 146,075 songs. Finally, we removed songs for which audio features were missing (i.e., had no value), resulting in a total of 142,963 songs for our final dataset. Those songs were performed by 25,667 distinct artists. Table 2 features an overview of the dataset.

4 Methods

Our approach to hit song prediction relies on a pairwise model which, given a pair of songs A and B, attempts to predict whether song A is more popular than song B. This approach is grounded in research on learning to rank [Li11], where pairwise models have superseded pointwise approaches. We propose a neural network model that takes the audio features (cf. Section 3.2) of two songs as its input and produces a binary label answering the above question as output. Further, we present two techniques for augmenting the training data to further improve the model's performance. In the following, we present the model, followed by our approaches toward training data augmentation.

4.1 Pairwise LTR Model

The model we propose is a quite straightforward neural network architecture. We employ a feed-forward network with six hidden layers of sizes between 256 and 64 neurons, and an output layer with a single neuron. A schematic depiction of our network architecture is given in Figure 1. The hidden layers all use SELU activation, as proposed by Klambauer et al. [Kl17]. We chose SELU units because they showed the best results in preliminary experiments, where we tested them against ELU, ReLU and tanh activations.

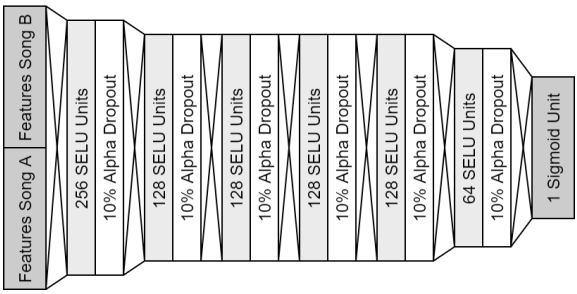

Fig. 1: A schematic depiction of our model.

As described by Klambauer et al. [Kl17], we initialized the weights of the network with values drawn from a normal distribution with zero mean and unit variance, as is necessary to obtain the self-normalizing property of SELU units. We also use the modified dropout variant proposed by Klambauer et al., with a dropout factor of 10%. For the output layer, we use a single neuron with sigmoid activation. The output is a value $y' \in [0, 1]$, which is then mapped to a label $y \in \{0, 1\}$, where $y = 1$ signifies that song A is more popular than song B, and $y = 0$ signifies that the opposite is the case. The mapping is done via

$$y = \begin{cases} 0 & \text{if } y' < 0.5 \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

4.2 Training Data Augmentation

To improve the classification performance of our model, we propose two augmentation strategies for the training data that aim to force the model to focus on those aspects that potentially make a song more popular than another. Note that these augmentation strategies are only possible due to the model being trained in a pairwise fashion—they could not be applied to a pointwise model. In the following, we explain those strategies and our reasoning behind them. Note that these augmentation strategies work independently of each other and always operate on the base dataset. In other words, samples generated by one augmentation step are not used as the input to another step.

4.2.1 Mirrored Training

The first augmentation strategy we propose is *mirrored training*. It works as follows: If our training dataset contains a pair of songs $x = (A, B)$ with label y, we add the pair $x' = (B, A)$ with label y' as defined in the following equation to the training set:

$$y' = \begin{cases} 0 & \text{if } y = 1 \\ 0 & \text{if } y = 0 \text{ and A and B have same play count} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

In other words, we mirror the pair of songs and invert the label (or leave it as 0, if both songs have the same play count, meaning that none is more popular than the other). We expect this augmentation strategy to help the model with learning to work independently of the order of the two input songs.

4.2.2 Reflexive Training

The second augmentation strategy we propose is *reflexive training*. It works as follows: If our training dataset contains a pair of songs $x = (A, B)$, regardless of the label, we add the following two instances to the training data:

- (A, A), with label $y = 0$
- (B, B), with label $y = 0$

In other words, we add pairs for both songs compared with themselves. The label is 0 in both cases since no song can be more popular than itself. Our reasoning for this strategy is that it should force the model to focus on *comparing* the two songs to determine which one is the more popular one, instead of focusing on the features of one of the songs. Without this augmentation, we expect that the model could, for example, simply learn to classify whether the first song in a pair is popular, essentially ignoring the second song.

5 Experiments

Given our model and dataset, we performed the following experiments to evaluate the performance of our pairwise model and the effectiveness of our augmentation strategies against a traditional pointwise model.

5.1 Generating Pairs of Songs

To obtain training and testing data for our pairwise model, we have to draw random pairs of songs from our dataset that we described in Section 3. For this, we first split our dataset

into training (64,175 songs), test (47,178 songs), and validation set 31,610 songs). The validation set was only used for preliminary experiments. For training and testing, we took the training and test sets, respectively, and then generated pairs of songs from each set as follows. Suppose we want to draw n pairs. We then generate two lists of length n of random integers in the range $[1, m]$ with $m = 142,963$ being the number of songs in the set. Those two lists are then used as the indices of the first and second song in the pairs, respectively. This means that if, for example, the two lists were $[1212, 2342, 1, 42, ...]$ and $[96, 232, 3636, 5, ...]$, the first pair would consist of songs number 1212 and 96, the second pair of songs 2342 and 232, etc. For the labels, we use the Last.fm play count of song X, $\mathrm{pc}(X)$, and for pair (A, B) set y as follows:

$$y = \begin{cases} 1 & \text{if } \mathrm{pc}(A) > \mathrm{pc}(B) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

5.2 Experiments for the Pairwise Model

To verify the performance of our pairwise model and the effectiveness of our augmentation strategies, we performed a set of four experiments:

- **pairwise**: An experiment with our model as described in Section 4, without using any of our augmentation strategies.
- **pairwise-mirror**: An experiment with the proposed model, but using the *mirrored training* augmentation strategy.
- **pairwise-reflexive**: An experiment with the same model, but using the *reflexive training* augmentation strategy.
- **pairwise-full**: An experiment with the same model, and using both the *mirrored training* and *reflexive training* augmentation strategies.

For our experiments, we drew 300,000 pairs of random songs for the training and 100,000 pairs for the test set. The random seed was fixed, so that all experiments were performed with the same base set of song pairs. For testing, we also performed mirroring augmentation if the model was trained with mirroring. We never performed reflexive augmentation for the test set, as in a practical application the model would never be used to predict if a song is more popular than itself. For all experiments, our model was trained using binary cross-entropy loss using the `Adam` optimizer [KB14]; we trained for 100 epochs.

5.3 Comparison: Pointwise Model

To judge how well our pairwise model performs compared to traditionally trained pointwise models for hit song prediction, we also performed experiments with a model trained in a

Model	Mirrored Training	Reflexive Training	F_1	Precision	Recall
pairwise-full	yes	yes	**0.670**	0.622	**0.726**
pairwise-mirror	yes	no	0.640	0.637	0.642
pairwise-reflexive	no	yes	0.626	**0.653**	0.602
pairwise	no	no	0.617	0.633	0.602
pointwise	n/a	n/a	0.629	0.627	0.631

Tab. 3: The results of our experiments.

traditional way—i.e., as a pointwise regression model predicting the play count of a given song. For this, we used the same general model architecture as for our pairwise model, to make results as comparable as possible, but reduced the input layer to only contain the features for a single song, as opposed to a pair of songs. For this experiment, the output layer activation function was removed, since we need the network to be able to output arbitrary values for predicting the play count of a given song. For training and testing, we used the same training and test set splits as outlined in Section 5.1. The training was done over 100 epochs, using mean squared error loss. For the evaluation, we randomly drew 100,000 pairs of songs from the testing set, using the same procedure as described in Section 5.1, and then used the model to predict the play counts for both songs in the pair. If the predicted play count of the first song was higher than that of the second song, we set the predicted label to 1, meaning that the model predicted the first song to be more popular, and otherwise we set it to 0. This procedure allows us to evaluate how well a pointwise model performs relative ranking—i.e., detecting which of two songs should be the higher ranked one—and makes it possible to compare the results of the pointwise and the pairwise model.

6 Results and Discussion

For the evaluation, we use the F_1 score, precision and recall metrics, since those metrics are widely used to measure the performance of classification models and provide a fair evaluation even if a dataset is not balanced, which should make our results more easily comparable to others. The results of the evaluation are given in Table 3. Looking at the pairwise models, a clear difference can be seen between the models using augmentation and the one which does not. The model without any form of data augmentation achieves an F_1 score of 0.617, which is outperformed by all models using augmentation. The models using only a single augmentation strategy show slight to moderate improvements, with an F_1 score of 0.626 for the model using reflexive training, and 0.640 for the model using mirrored training. Notably, mirrored training seems to primarily boost recall, improving from 0.602 for the model without augmentation to 0.642, while reflexive training seems to primarily boost precision, improving from 0.633 for the model without augmentation to 0.653. Combining both augmentation strategies leads to the best performance, with an F_1 score of 0.670. This shows that the proposed augmentation methods are individually effective and complement each other well.

The pointwise model achieves an F_1 score of 0.629. Comparing to the pairwise models, we see that while the pointwise model perform better than the pairwise model trained without any augmentation, the models using a single augmentation strategy perform about as good as the regression models. The model combining both augmentation strategies clearly outperforms the regression models, with an F_1 score of 0.670 compared to 0.629. This shows that training a model in a pairwise fashion and leveraging suitable augmentation strategies—which only becomes possible due to the model being a pairwise model—leads to a model that is better at predicting the relative popularity of two songs, at least in our setup.

We do note, however, that there are limitations to our results. First, while we tried to keep the bias towards more popular songs in our dataset as low as possible, we acknowledge that our data sources most probably still exhibit such a bias. Nonetheless, the popularity in our dataset shows a characteristic long tail distribution, implying that this bias is small. Second, our dataset is also biased towards slightly older songs, containing no songs released after January 2015. We note that, in the context of using play counts as a popularity measure, this, in fact, helps reduce a popularity bias in the data, as older songs naturally had more time to accrue plays, which would unfairly disadvantage newer songs.

7 Conclusion

In this paper, we proposed a pairwise approach for hit song prediction. We created a model which, given a pair of songs A and B, predicts whether song A is more popular than song B. For this, we proposed a neural network architecture which takes as its input audio features of two songs and thus can be trained to directly answer that question, as opposed to traditional pointwise approaches for popularity prediction. Furthermore, we proposed two data augmentation strategies to improve the performance of our model. Our results showed that both augmentation strategies are effective in improving the performance of our pairwise model, and that using both together leads to the best results. We also performed a comparison with a model trained in a traditional way—i.e., as a pointwise model—and showed that our pairwise model trained using both of our augmentation strategies clearly outperforms it.

Future work includes analyzing the importance of individual features, to determine which features allow a pairwise model to determine which song is more popular. Another possible direction for future work is to investigate whether other popularity measures (distinct listener count, chart positions, etc.) can also effectively be used with a pairwise model. As our results suggest that approaches from learning to rank seem to be transferable to hit song prediction, a natural next step would also be to use more sophisticated pairwise learning to rank models to predict the popularity of songs.

Bibliography

[Bo13a] Bogdanov, Dmitry; Wack, Nicolas; Gómez, Emilia; Gulati, Sankalp; Herrera, Perfecto; Mayor, Oscar; Roma, Gerard; Salamon, Justin; Zapata, José; Serra, Xavier: ESSENTIA: An Open-Source Library for Sound and Music Analysis. In: Proc. of the 21st ACM International Conference on Multimedia. pp. 855–858, 2013.

[Bo13b] Bogdanov, Dmitry; Wack, Nicolas; Gómez Gutiérrez, Emilia; Gulati, Sankalp; Herrera Boyer, Perfecto; Mayor, Oscar; Roma Trepat, Gerard; Salamon, Justin; Zapata González, José Ricardo; Serra, Xavier: Essentia: An Audio Analysis Library for Music Information Retrieval. In: Proc. of the International Society for Music Information Retrieval Conference. pp. 493–498, 2013.

[DL05] Dhanaraj, Ruth; Logan, Beth: Automatic Prediction of Hit Songs. In: Proc. of the International Society for Music Information Retrieval Conference. pp. 488–491, 2005.

[KB14] Kingma, Diederik P; Ba, Jimmy: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[Kl17] Klambauer, Günter; Unterthiner, Thomas; Mayr, Andreas; Hochreiter, Sepp: Self-normalizing neural networks. In: Proc. of the 31st International Conference on Neural Information Processing Systems. pp. 972–981, 2017.

[Li11] Liu, Tie-Yan: Learning to Rank for Information Retrieval. Springer Science & Business Media, 2011.

[LL18] Lee, Junghyuk; Lee, Jong-Seok: Music Popularity: Metrics, Characteristics, and Audio-Based Prediction. IEEE Transactions on Multimedia, 20(11):3173–3182, 2018.

[Ni11] Ni, Yizhao; Santos-Rodriguez, Raul; Mcvicar, Matt; De Bie, Tijl: Hit Song Science Once Again a Science. In: 4th International Workshop on Machine Learning and Music. 2011.

[PR08] Pachet, François; Roy, Pierre: Hit Song Science Is Not Yet a Science. In: Proc. of the International Society for Music Information Retrieval Conference. pp. 355–360, 2008.

[Sc16] Schedl, Markus: The lfm-1b dataset for music retrieval and recommendation. In: Proc. of the 2016 ACM on International Conference on Multimedia Retrieval. pp. 103–110, 2016.

[Ya17] Yang, Li-Chia; Chou, Szu-Yu; Liu, Jen-Yu; Yang, Yi-Hsuan; Chen, Yi-An: Revisiting the Problem of Audio-based Hit Song Prediction Using Convolutional Neural Networks. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 621–625, 2017.

[Yu17] Yu, Lang-Chi; Yang, Yi-Hsuan; Hung, Yun-Ning; Chen, Yi-An: Hit song prediction for pop music by siamese CNN with ranking loss. arXiv preprint arXiv:1710.10814, 2017.

[Za19] Zangerle, Eva; Vötter, Michael; Huber, Ramona; Yang, Yi-Hsuan: Hit Song Prediction: Leveraging Low- and High-Level Audio Features. In: Proc. of the International Society for Music Information Retrieval Conference. 2019.