Text-Based Classification of Song Lyrics by a single Artist

Nikolaus Stratil-Sauer
Linköpings universitet
nikst363@student.liu.se

Abstract

This project uses text classification to classify music lyrics of an artist to targets concerning that artist, like albums or phases in their career. For this 93 songs of the metal band Nightwish were classified. This is a unique challenge given the low amount of training data. The best classification models consisted of pretrained word embeddings using BERT and a Naive Bayes or Logistic Regression classifier. For three different classification targets (classifying songs to albums(9), lead singers(3) and eras(2) of the band), accuracies between 50% and 78% were achieved. Other classifiers based on TF-IDF Vectorizers or on a convolutional neural network showed performances not much better than the baseline.

1 Introduction

Lyrics of songs are a big source of contemporary discussion. They can capture social and political themes of the time and as such are interesting to analyse in the context of text mining. Text classification methods have been used to determine the sentiment or mood of a song (Hu et al., 2009), to classify lyrics into different music genres (Tsaptsinos, 2017) and even to differentiate between artists of the same genre (Bužić and Dobša, 2018), showing good results for these tasks. One area that remains unexplored to our knowledge, is the task of using song lyrics of a single artist to perform classification for aspects related only to that artist.

In this project we aim to classify lyrics of songs of the symphonic metal band Nightwish. Nightwish is one of the biggest metal bands of Europe and known as the founder of the symphonic metal genre. In their 28-year long history, the band has released 9 albums, with a total of around 100 songs (though some songs are excluded from this project as they are instrumentals or only contain a few lines of lyrics). Over the course of time, the lyrical themes in their songs have shifted substantially. Earlier albums contained a lot of references to mythology and fantasy (see the song "Wishmaster" on the third album [1]), while newer releases are focused on humanity and human nature (see the song "Noise" from the latest album [2]).

For this project we identified three classification goals. The first is to classify the songs according to the album they were released in. The other goals refer to the progression Nightwish made over the course of their career. After the first five albums, the band switched their lead singer from Tarja Turunen first to Anette Olzon and then after their seventh album Floor Jansen became the new lead singer. With their first switch to Anette Olzon it can also be argued that their musical style evolved slightly from a more power metal focused style to symphonic metal. Thus the other two classification goals are to classify song lyrics according to their singer and according to their era (Tarja Turunen and Post-Tarja Turunen). The era grouping is slightly subjective of course, but nonetheless it is interesting to see if text classification methods can learn a difference between these groups and accurately predict the group memberships of a previously unseen song, just from the lyrics.

2 Data

The dataset was scraped from the offical Nightwish website[3], where all song lyrics are available. Each data record was manually extended by additional information according to the classification goals (album, singer and era). In total this results in 93 songs from 9 different albums. As we can see in Figure 1 the number of songs per album is quite balanced and lies between 9 and 12.

The data for our other classification goals is distributed as follows. The songs are approximately evenly distributed between the two different eras of Nightwish. The first era contains 51 songs, while

[1] https://www.nightwish.com/songs/wishmaster
[2] https://www.nightwish.com/songs/noise
[3] https://www.nightwish.com/music

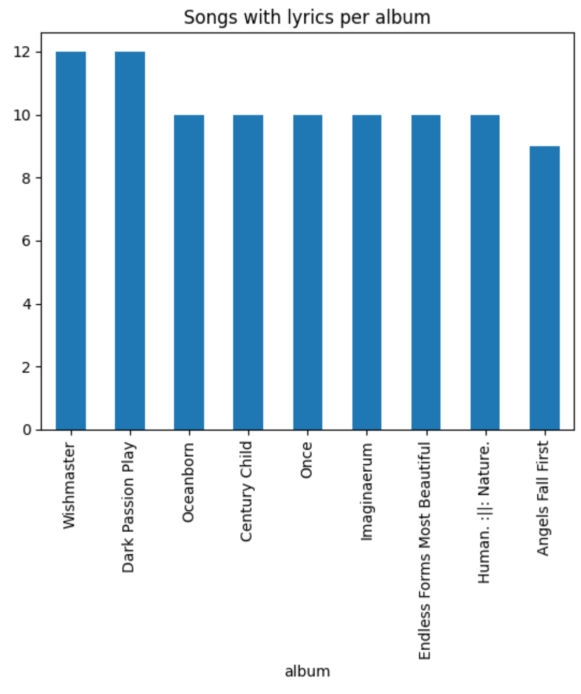

Figure 1: Number of songs in the dataset per album

the second contains 42 songs. For the singers the distribution is more unbalanced. Tarja Turunen sang for 51 songs, Anette Olzon for 22 songs and Floor Jansen for 20 songs. This makes it necessary to take extra precautions when splitting the data into a training and test data set and also when choosing a metric for comparing the models. The details of this process are explained in the next section.

3 Method

In order to classify the song lyrics according to our classification goals, we compare a number of vectorizers, embeddings and classifiers. The methodology is explained in this section. Note that the CNN used in this project uses both a embeddings and a classifier but it is explained in a separate sub-section, due to its complexity.

3.1 Train-test split

As a first step it is necessary to split our obtained dataset into a training and a test set. A possible validation set is omitted and instead we use cross validation to compare different models. This is preferable due to the low total number of data records. We want to preserve as many records as possibly for training the models. Only the final selected model for each classification goal is tested on the test set, in order to gauge its performance on unseen data. The training to test ration is chosen at roughly 80 vs. 20%. Through adapted (manual) stratified splitting, this results in 2 songs of each album being in the test set for the album classification, 9 songs of each era for era classification and 8 songs of Tarja Turunen and 5 songs of the other singers, for the unbalanced singer classification set.

3.2 Text vectorization and word embedding

In order to use classifiers on our text data we need to bring it into a format that can be used by our classification methods. For this several approaches exist. On the one hand we use simple methods like bag-of-words vectors and TF-IDF vectorizers. These only look at the frequency of words appearing in the documents and in the data set as a whole. The vectorizers used in this project are the `CountVectorizer` and the `TfidfVectorizer` from the `scikit-learn` package. In this project we compare the results of vectorizers using both unigrams and n-grams of the range (2,3) in order to get a more complete picture.

Another approach is to use pretrained embeddings. These transfer words into vectors of real numbers and can capture semantics and meanings behind words. This can capture additional context and thus improve classification performance. In this project we use embeddings from BERT using the "bert-base-uncased" [4] model. For each document BERT provides the classifying token "[CLS]", which is pretrained for the use in sentence-level classification tasks. We use this token as our embedding of each lyric. In order to perform the embeddings, we use the `AutoModel` and `AutoTokenizer` classes from the `transformers` package. BERT has an upper limit of 512 tokens per sentence, which one of our lyrics reaches, as it has 743 tokens. In this case the lyric is automatically truncated to comply with the maximum length.

We do not remove stop words or treat the text in any other way before computing vectors. Previous studies classifying music lyrics have found that doing this could actually decrease the eventual classification performance (Howard et al., 2011).

3.3 Classification

In order to maximize classification performance, several classifiers were trained on the training data set. They are trained using `cross_validate` func-

[4]https://huggingface.co/bert-base-uncased

tion from the package `scikit-learn`. This function performs 5-fold cross-validation and uses stratified splitting to maintain the original ratio between the groups in the training data. The average validation accuracy of these five folds is the metric we use to compare the models. The used classifiers are implemented using the `scikit-learn` package as well. These are the classifiers in this project:

1. Multinomial Naive Bayes
2. Support Vector Machine (with the parameter `gamma = "auto"`)
3. Logistic Regression (with the parameter `max_iter = 1000`, as otherwise the algorithm sometimes fails to converge)
4. RandomForestClassifier. In order to find the optimal parameters `max_depth` and `min_sample_split` we use grid search via the `GridSearchCV` function.

These classifiers are trained on our three classification goals and on the different vectorizations and embeddings separately. To compare them we chose the accuracy metric for the album and era classification goals. As the singer classification features a very unbalanced data set, comparing by accuracy alone is not enough to determine the best model. Thus we also take the F1 metric into account for this classification goal.

3.4 CNN

In recent years Neural Networks have proven to be able to outperform traditional methods in the task of text classification (Minaee et al., 2021). In this project we chose to use the sepCNN architecture (Chollet, 2017) as it provides a good balance between performance and computational complexity according to sources[5] comparing multiple RNN and CNN architectures in the task of text classification. SepCNN is a CNN where standard 3D convolutions are factored into two more computationally efficient convolutions, a depthwise convolution and a pointwise convolution. This process is called separable convolution. These convolutions are followed by a pooling layer. In order to guard against overfitting, a dropout layer is included. The CNN is implemented using the `keras` python library. For more details on the precise implementation, refer to the code repository and the python notebook "sepCNN" [6].

A CNN also needs to vectorize the text data in some way. For this two possibilites exist. The first is to include an empty `Embedding` layer and let the model train the embeddings itself. The other one is to refer to pretrained embeddings, as a starting point. These pretrained embeddings are trainable by the model to let it adapt the pretrained embeddings to our context of music lyrics. The embeddings we use in this case stem from the GloVe algorithm (Pennington et al., 2014). Specifically, we use the Glove 6B 300d vectors, trained on a Wikipedia dataset [7].

To be able to use cross validation as we used it previously with the other classifiers, we use the `scikeras` package with the `KerasClassifier` function to port the CNN model into a `scikit-learn` classifier. The sepCNN structure includes a number of hyperparameters. Due to the expensive computation an exhaustive grid search to find the optimal configuration is not in the scope of this project. Instead we fine-tuned the hyperparameters manually. The best balance between over- and underfitting was found using 200 epochs, a learning rate of 0.015 and a dropout rate of 0.4.

4 Results

Due to the large number of results, refer to the appendix for detailed accuracy scores of each vectorizer - classifier combination. See Table 1 for results for classification for albums and eras and Table 2 for the results of the classification for singers. In this section we shortly present an overview over the results for each classification goal and present the test accuracy and confusion matrix of the best model.

4.1 Classifying Albums

For classifying albums, using word embeddings seems to deliver the best results. Embeddings and the Naive Bayes algorithm reach the highest accuracy at 0.320, with other classifiers reaching similar values on the embeddings. The bag-of-words vectorizer, utilizing n-grams also stands out with a maximum accuracy of 0.280. TfIdf vectorizers using n-grams and also the sepCNN algorithm

[5]https://developers.google.com/machine-learning/guides/text-classification/step-4

[6]https://github.com/nikostra/nighwish-classification

[7]https://nlp.stanford.edu/projects/glove/

are struggling in this task with accuracies that are only slightly higher than that of a dummy classifier choosing classes at random (0.11).

The combination of word embeddings and Naive Bayes classifier reaches a 0.5 accuracy on our test set. This is considerably higher than the validation accuracy. A possible reason for this could be easier to classify examples in the test set, as due to the low total number of data records and the varying difficulty of the songs, training and test sets might not be balanced perfectly.

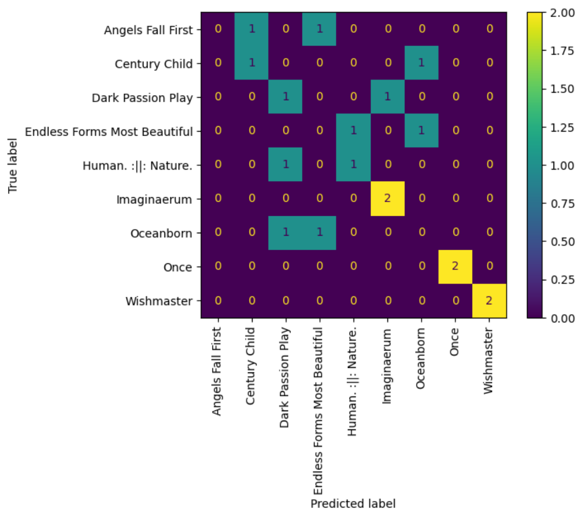

Figure 2: Confusion matrix of the album classification on the test set.

In Figure 2 we can see that some of the albums get classified precisely, while the algorithm struggles to learn others. The album "Angels Fall First" never gets predicted, indicating that this album could be particularly hard to predict or that it features many different lyrical themes.

4.2 Classifying Eras

The results for predicting the right Nightwish-era are considerably better than when classifying albums. Using a dummy classifier that predicts the biggest class gives us a baseline accuracy of 0.548 in this case. The best actually obtained accuracy is 0.787 and comes from the logistic regression classifier using word embeddings. Again the word embeddings prove more accurate than the other methods, as all classifiers using word embeddings show high accuracies of 0.733 and above. The CNN struggles and shows only an accuracy that is barely higher than the baseline. In this example the bag-of-words vectorizer using unigrams together with a naive bayes classifier also provides a reasonable performance at 0.733 accuracy.

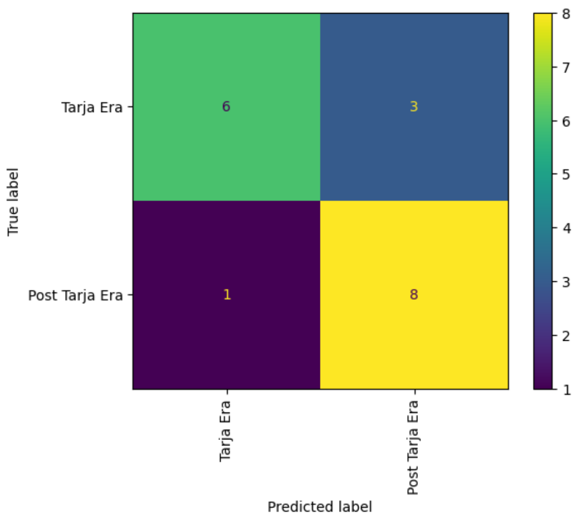

Figure 3: Confusion matrix of the era classification on the test set.

The test accuracy of our best model lies at 0.78, which is comparable to our validation accuracy. The confusion matrix in Figure 3 shows that the model slightly overpredicts songs to the Post-Tarja era.

4.3 Classifying Singers

For this classification task we could compare against two different baselines. The first would be a classifier predicting a random singer giving an accuracy of 0.333. The other option is a classifier always predicting the most frequent singer, which is Tarja Turunen at 51 songs giving us a theoretical accuracy of 0.548, but a considerably lower average F1 score, as the F1 score for the other classes would be 0 in this case.

This classification goal shows again that using word embeddings seems to be the best method for this dataset. Together with a logistic regression classifier they achieve 0.747 accuraccy and F1 mean of 0.7. The random forest classifier using word embeddings achieves an even higher validation accuracy at 0.773, but a lower F1 score. Performances from the CNN and from the other vectorizers are largely disappointing with performance metrics that don’t significantly improve on the baseline scores. The bag of words vectorizer together with a Naive Bayes classifier performs best out of these.

Using the logistic regression classifier together with word embeddings on our test set gives an accuracy of 0.56 and a F1 score of 0.52. This is a

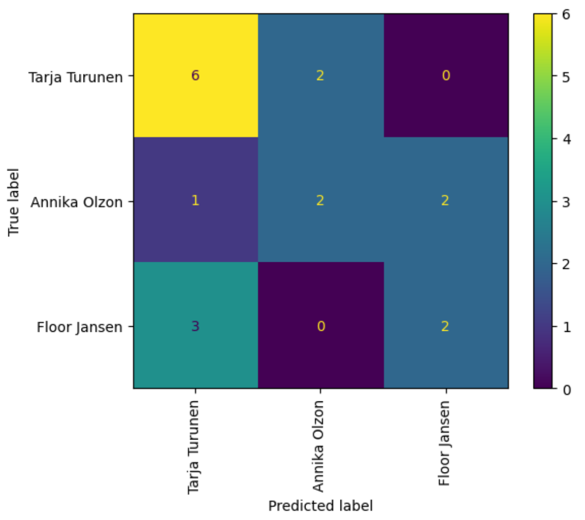

Figure 4: Confusion matrix of the singer classification on the test set.

disappointing result as it is considerably lower than our validation metrics and in range of the baseline performance. We can see in the confusion matrix that the model struggles to accurately predict songs to the singers Annika Olzon and Floor Jansen. Perhaps models are struggling with learning these groups, as there are less examples of them in the training data. This does not really explain the big difference between validation and test accuracy however.

5 Discussion

As we saw in the previous section, we could find classifiers for each classification goal that significantly outperformed baseline classifiers. That being said, especially for the goal of classifying albums, the performance is not satisfactory. In a real world scenario our classifier would classify every second or third song to the correct album, which would not be enough in most use cases. Figure 5 shows a visualization of the word embeddings of all songs using the t-SNE method (Van der Maaten and Hinton, 2008). We can see that while there is some overlap between albums (especially in the Human. :II: Nature. album), for most albums it is hard to see clear decision boundaries. Considering the limited data availability it is reasonable and expected that classifiers struggle with this task. Classifying eras and singers shows more promise on the other hand and we achieved decent metrics in these tasks. It has to be mentioned though that while the validation accuracy for predicting singers is very decent, the test accuracy for this target is considerably lower due to unknown factors.

In general word embeddings proved to be the best method to vectorize the text data, even though simpler bag-of-words vectors were not far behind in some cases. As for the classifiers, all four selected options (Naive Bayes, Random Forest, Logistic Regression and SVM) showed largely similar performance, with Naive Bayes and Logistic Regression being the most consistent methods. The Random Forest classifier can be powerful, but it also showed a lot of variance inbetween runs, due to the randomness inherent in the algorithm and our small train dataset.

The CNN classifiers showed disappointing results and could not compete with the other methods in accuracy. While neural networks are a powerful tool in many contemporary applications and also in the context of text classification, for this task the limited data availability posed too many challenges. CNNs are complex structures with lots of trainable parameters and tunable hyperparameters. Finding the best setup for these, requires more training data than what is available in this project.

6 Conclusion

In this project we used text classification to classify music lyrics of an artist according to goals like albums or phases in the artists development. Lyrics of the symphonic metal band Nightwish were analysed. Using pretrained word embeddings and a Logistic Regression or Naive Bayes classifier we achieved an accuracy of 0.78 for classifying eras and 0.56 for classifying lyrics to one of the three lead singers of Nightwish. Classifying lyrics into one of the 9 albums of Nightwish resulted in a 0.50 accuracy on test data.

While other related research has focused on classifying music lyrics into different genres or artists, this project shows that is also possible to classify for goals within an artist with decent accuracy. Nevertheless, the low data availability poses problems for this type of classification and the results, especially for classifying albums and partially also for singers are not as good as they could be. For artists with even fewer songs than Nightwish, classification accuracies are expected to go down even more. On the other hand it would be interesting to see the results of a dataset consisting of multiple artists where we want to classify for specific albums for example.

To improve the performance several improve-

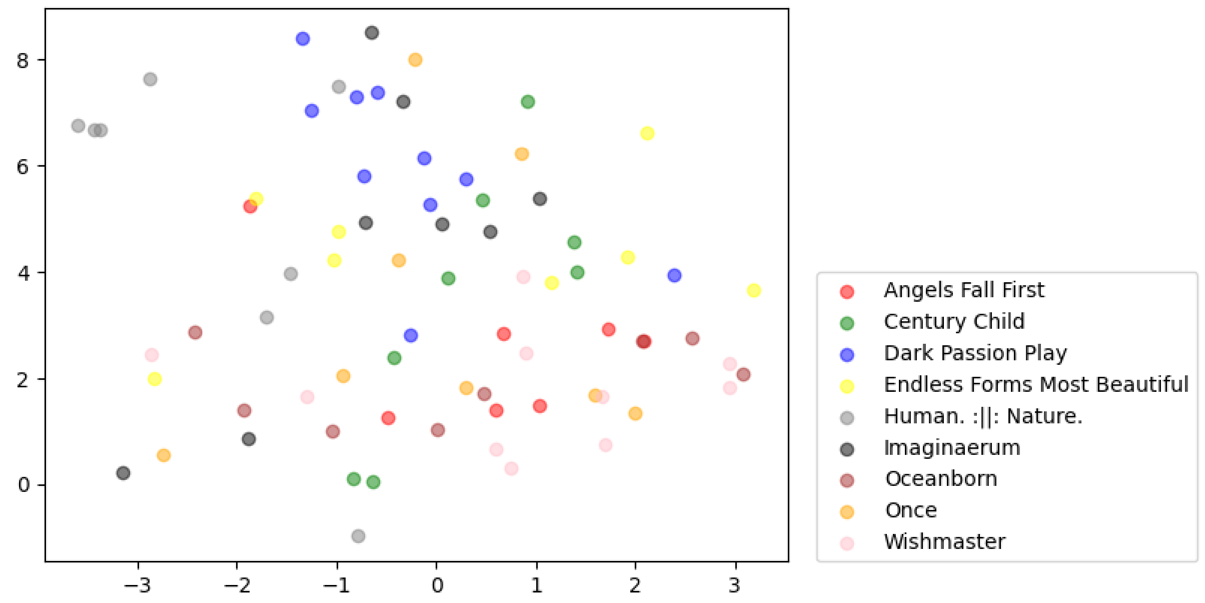

Figure 5: t-SNE plot of songs per album.

ments could be made in future research. It is possible to extend the text data with more features, for example employing topic modelling to provide even more context to the classifiers and group similar songs. Another possibility would be to use audio features as a feature for classification, as other projects have done for music genre classification (Elbir and Aydin, 2020).

References

Dalibor Bužić and Jasminka Dobša. 2018. Lyrics classification using naive bayes. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1011–1015. IEEE.

F. Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, Los Alamitos, CA, USA. IEEE Computer Society.

Ahmet Elbir and Nizamettin Aydin. 2020. Music genre classification and music recommendation by using deep learning. *Electronics Letters*, 56(12):627–629.

Sam Howard, Carlos N Silla Jr, and Colin G Johnson. 2011. Automatic lyrics-based music genre classification in a multilingual setting. In *Proceedings of the Thirteenth Brazilian Symposium on Computer Music*.

Xiao Hu, J Stephen Downie, and Andreas F Ehmann. 2009. Lyric text mining in music mood classification. *American music*, 183(5,049):2–209.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. *arXiv preprint arXiv:1707.04678*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Appendix

Model	**Accuracy (Album)**	**Accuracy (Era)**
CountVectorizer (unigrams) + Naive Bayes	0.267	0.733
CountVectorizer (unigrams) + Logistic Regression	0.213	0.72
CountVectorizer (unigrams) + Random Forest	0.213	0.613
CountVectorizer (unigrams) + SVM	0.147	0.653
CountVectorizer (ngram_range = (2,3)) + Naive Bayes	0.280	0.640
CountVectorizer (ngram_range = (2,3)) + Logistic Regression	0.200	0.547
CountVectorizer (ngram_range = (2,3)) + Random Forest	0.160	0.547
CountVectorizer (ngram_range = (2,3)) + SVM	0.147	0.547
TfIdfVectorizer (unigrams) + Naive Bayes	0.160	0.573
TfIdfVectorizer (unigrams) + Logistic Regression	0.267	0.693
TfIdfVectorizer (unigrams) + Random Forest	0.240	0.640
TfIdfVectorizer (unigrams) + SVM	0.200	0.547
TfIdfVectorizer (ngram_range = (2,3)) + Naive Bayes	0.147	0.547
TfIdfVectorizer (ngram_range = (2,3)) + Logistic Regression	0.147	0.547
TfIdfVectorizer (ngram_range = (2,3)) + Random Forest	0.120	0.520
TfIdfVectorizer (ngram_range = (2,3)) + SVM	0.173	0.547
Word embeddings + Naive Bayes	**0.320**	0.733
Word embeddings + Logistic Regression	0.280	**0.787**
Word embeddings + Random Forest	0.300	0.773
Word embeddings + SVM	0.2933	0.760
sepCNN with pretrained embeddings	0.120	0.573
sepCNN with learnable embeddings	0.133	0.573

Table 1: Validation metrics for albums and eras. The best models are highlighted in green.

Model	**Accuracy (Singer)**	**F1 - score (Singer)**
CountVectorizer (unigrams) + Naive Bayes	0.613	0.441
CountVectorizer (unigrams) + Logistic Regression	0.627	0.488
CountVectorizer (unigrams) + Random Forest	0.560	0.273
CountVectorizer (unigrams) + SVM	0.547	0.236
CountVectorizer (ngram_range = (2,3)) + Naive Bayes	0.613	0.478
CountVectorizer (ngram_range = (2,3)) + Logistic Regression	0.547	0.236
CountVectorizer (ngram_range = (2,3)) + Random Forest	0.547	0.236
CountVectorizer (ngram_range = (2,3)) + SVM	0.547	0.236
TfIdfVectorizer (unigrams) + Naive Bayes	0.547	0.236
TfIdfVectorizer (unigrams) + Logistic Regression	0.547	0.236
TfIdfVectorizer (unigrams) + Random Forest	0.560	0.273
TfIdfVectorizer (unigrams) + SVM	0.547	0.236
TfIdfVectorizer (ngram_range = (2,3)) + Naive Bayes	0.547	0.236
TfIdfVectorizer (ngram_range = (2,3)) + Logistic Regression	0.547	0.236
TfIdfVectorizer (ngram_range = (2,3)) + Random Forest	0.547	0.236
TfIdfVectorizer (ngram_range = (2,3)) + SVM	0.547	0.236
Word embeddings + Naive Bayes	0.707	0.632
Word embeddings + Logistic Regression	**0.747**	**0.700**
Word embeddings + Random Forest	0.773	0.663
Word embeddings + SVM	0.707	0.553
sepCNN with pretrained embeddings	0.520	0.284
sepCNN with learnable embeddings	0.573	0.243

Table 2: Validation metrics for singers. The best models are highlighted in green.