ICSI in MediaEval 2017 Multi-Genre Music Task

Kijung Kim[1], Jaeyoung Choi[2]
[1]University of California, Berkeley, CA, United States
[2]International Computer Science Institute, Berkeley, CA, USA
kijung@berkeley.edu,jaeyoung@icsi.berkeley.edu

ABSTRACT

We present our approach and result for the MediaEval 2017 AcousticBrainz Content-based music genre recognition task. Experimental results show that the best results come from random forest with partial feature selection.

1 INTRODUCTION

The 2017 Content-based music genre recognition from multiple sources Task [1] consists of two subtasks: single source classification and multiple source classification. We focused on the first subtask, which the goal was to predict genres using a single source of ground truth with broad genre categories as class labels. In the following sections, we describe our feature formulation, models and experiments in details.

2 TECHNICAL APPROACH

The proposed framework can be divided into three phases: (1) feature formulation , (2) standardization, and (3) model selection and predictions.

(1) **Feature Formulation** The dataset provides each song with three groups of pre-extracted features: tonal, rhythm, and low-level. A feature vector for each song was formed as a concatenation of all the individual features from each group. For features with specifics labels such as mean, max, and min, they were simply concatenated together. For the sake of simplicity, categorical features were not considered. The excluded features are: "key_key", "key_scale", "chords_key", and "chords_scale". The "beats_position" was excluded as the feature for each song has variable length, and we assumed that the features "bpm" and "beats counts" were sufficient. This resulted in a 2647-dimensional feature vector for each song.

(2) **Standardization** We randomly sampled a subset of 100,000 songs for each dataset, formulated the feature vector, and computed the mean and standard deviation for all indicies in the feature vector. Then, at the test phase, each feature was standardized using the pre-computed mean and standard deviation.

(3) **Model Selection and Predictions** From scikit-learn [6], two classifiers used in our approach were the Stochastic Gradient Descent (SGD) classifier with hinge loss and Random Forest classifier with 16 estimators [2]. A binary classifier was trained for each genre/subgenre, the results were conglomerated together and prediction for each genre/subgenre was made independently.

The first two Runs consisted of concatenating all provided features (except the ones mentioned above) and using the SGD classifier.

MediaEval'17, 13-15 September 2017, Dublin, Ireland

2.1 Stochastic Gradient Descent Classifier: Run 1 and 2

Run 1 consisted of each song having a concatenated feature vector of all features minus the ones mentioned above with the SGDClassifier. To accommodate for large data, batch training of size 80,000 was used.

Run 2's feature formulation and the model is the same as Run 1. The difference is in the prediction process. The procedure was to look at the results for each song and mainly go with the genre prediction. For example, given main genre A has subgenres B,C and main genre D has subgenres E, F, if the classifiers classified a song as main genre A with subgenres C,D, and F but does not classify it as main genre D, because main genre D was not predicted, the predictions will ignore subgenre F, and the final prediction will be genre A with subgenres C,D. In short, subgenre predictions were ignored if their main "parent" genres were not predicted. This approach was to decrease the chance of false positives for subgenres. In short, we made a system of hierarchy and weighed genre predictions higher than subgenre predictions.

2.2 RandomForest with Parital Feature Selection: Run 3, 4 and 5

For the next three Runs, we used random forest classifier (RFC) with partial feature selection. We used the feature importance [4] from the trained random forest classifier. We first took a subset of the train data (around 100,000 songs), formulated a concatenated feature vector for each song, and fit the features to the RFC for each genre and subgenre. Then, we used the ranked feature importance list from the classifier to select the x% best features, which resulted in different best features for each genre and subgenre [5]. From there, we trained all-for-one RFC's using the top x% features for each genre and subgenre with its own x% best features, and used a subset of the train data (around 150,000 songs). Finally, prediction of the genres were made based on a conglomeration of all the RFC's.

Run 3, 4, and 5 used the top 25%, 50%, and 75% of the features from the ranked list of feature importance from the trained RFC, that resulted in a 661, 1323, and 1985 dimensional feature vector per song, respectively.

3 RESULTS AND ANALYSIS

In this section, we report accumulated results on the sub-task based on our two different approaches.[1] Our results are reported in Figure 1. The test set is composed of three different databases (Discogs, Lastfm, Tagtraum), and we took the average of precision, recall, and f-score to obtain single number.

We observe that the approaches based on the Random Forest Classifiers (Runs 3, 4, 5) outperform the SGD Classifier approaches

[1]https://multimediaeval.github.io/2017-AcousticBrainz-Genre-Task/results/

Table 1: Subtask 1 Results

dataset	Run	Ptrackall	Rtrackall	Ftrackall	Ptrackgen	Rtrackgen	Ftrackgen	Ptracksub	Rtracksub	Ftracksub	Plabelall	Rlabelall	Flabelall	Plabelgen	Rlabelgen	Flabelgen	Plabelsub	Rlabelsub	Flabelsub
discogs	Run 1	0.0109	**0.65**	0.0214	0.0989	**0.791**	0.172	0.006	**0.5171**	0.0117	0.0104	**0.5702**	0.0163	0.0963	**0.7399**	0.1471	0.0061	**0.5617**	0.0098
discogs	Run 2	0.0136	0.6083	0.0264	0.0989	**0.791**	0.172	0.0069	0.4322	0.0135	0.0105	0.4496	0.0164	0.0963	**0.7399**	0.1471	0.0062	0.435	0.0099
discogs	Run 3	**0.032**	0.3337	**0.0568**	**0.2059**	0.4023	**0.2475**	**0.0181**	0.2735	**0.0331**	**0.0247**	0.2484	**0.0349**	**0.1375**	0.3149	**0.1638**	**0.0191**	0.2451	**0.0285**
discogs	Run 4	0.028	0.3621	0.0508	0.1725	0.4595	0.2326	0.0153	0.2823	0.0284	0.0191	0.2585	0.0293	0.1095	0.351	0.1409	0.0146	0.2538	0.0237
discogs	Run 5	0.0216	0.3493	0.0401	0.0999	0.3943	0.1513	0.0136	0.3092	0.0257	0.016	0.2764	0.025	0.0889	0.3908	0.117	0.0123	0.2707	0.0204
lastfm	Run 1	0.0065	**0.4746**	0.0129	0.0454	**0.4887**	0.0824	0.0048	**0.4981**	0.0096	0.0089	**0.5392**	0.0105	0.0553	**0.4076**	0.0466	0.0042	**0.5525**	0.0068
lastfm	Run 2	0.0097	0.3715	0.0187	0.0454	**0.4887**	0.0824	0.0054	0.2648	0.0106	0.0117	0.268	0.0089	0.0553	**0.4076**	0.0466	0.0073	0.2539	0.0051
lastfm	Run 3	**0.0385**	0.2619	**0.0637**	**0.0993**	0.3053	**0.1406**	**0.0274**	0.219	**0.0461**	**0.0307**	0.2075	**0.0406**	**0.0926**	0.2585	0.1026	**0.0244**	0.2023	**0.0343**
lastfm	Run 4	0.0338	0.2711	0.0575	0.0883	0.3338	0.1331	0.0219	0.202	0.0379	0.0264	0.1978	0.0353	0.0815	0.2831	**0.1049**	0.0208	0.1892	0.0282
lastfm	Run 5	0.0273	0.277	0.0483	0.0691	0.3284	0.1098	0.0191	0.2187	0.0341	0.0217	0.206	0.0311	0.0591	0.277	0.0795	0.0179	0.1989	0.0262
tagtraum	Run 1	0.0114	**0.5375**	0.0222	0.0372	0.3977	0.0673	0.0095	**0.6141**	0.0187	0.0129	**0.5859**	0.0185	0.0509	0.4937	0.0514	0.0084	**0.5967**	0.0146
tagtraum	Run 2	0.0129	0.3272	0.0246	0.0371	**0.3981**	0.0673	0.0082	0.2519	0.0157	0.0151	0.2915	0.018	0.0509	**0.494**	0.0513	0.011	0.2678	0.0141
tagtraum	Run 3	**0.0487**	0.2655	**0.0782**	0.1213	0.3366	0.1654	**0.0318**	0.2137	**0.0528**	**0.0384**	0.2099	**0.0488**	**0.111**	0.3349	**0.1327**	**0.0299**	0.1953	**0.039**
tagtraum	Run 4	0.0456	0.3201	0.0774	**0.1276**	0.4357	**0.1876**	0.0273	0.2432	0.0477	0.0326	0.2257	0.0443	0.0957	0.3291	0.1232	0.0253	0.2136	0.0351
tagtraum	Run 5	0.0336	0.2968	0.0589	0.0636	0.3376	0.1038	0.0259	0.2562	0.0458	0.0262	0.2324	0.0374	0.0722	0.3845	0.094	0.0208	0.2146	0.0307

(Runs 1, 2). In particular, we note that the recall in Runs 1, 2 is especially high while the precision is especially low, which meant the classifiers predicted each song with almost every label. For Runs 3, 4, and 5, we observe a significantly lower recall with a better precision.

For Runs 3, 4, and 5, we observe a trend that by adding additional features, the recall improves at the cost of precision. However, Run 5 disproves such trend, as it shows that Run 5 gives only better recall for per-label results while showing worse result in all metrics in per-track results.

4 CONCLUSION

Runs 1 and 2 clearly suffered from oversampling, which lead the classifiers in most genres to predict positive, which resulted in high recall and low precision. Runs 3,4, and 5 did not suffer like Runs 1 and 2, but upon observing precision, recall, and f-scores for each genre, the classifiers did far worse on non-popular genres and subgenres, which lead to overall lower precision and recall.

The shortcomings came, for Runs 1 and 2, from errors in sampling. These sampling errors were technical, as they originated from code. For Runs 3,4, and 5, the shortcomings came from a lack of a system to combine results from different classifiers. For one, we could have exploited the probabilities generated from the model for each prediction to ascertain a threshold for each genre and subgenre. This would have helped especially for sparse subgenres.

For future works, for Runs 1, 2, it would be interesting to see if taking less top % of the features from the feature importance list for each genre and subgenre will improve precision. Also, it may be worth trying majority voting by training several different random forest classifiers using the feature importance list. Lastly, it would be interesting to try the imbalanced-learn package [3] which is compatible with scikit-learn and may fix the class imbalance for Runs 1 and 2.

ACKNOWLEDGMENTS

This work was supported in part by AWS Research Grants.

REFERENCES

[1] Dmitry Bogdanov, Alastair Porter, Julian Urbano, and Hendrik Schreiber. 2017. The Mediaeval 2017 AcousticBrainz Genre Task: Content-based music genre recognition from multiple sources. *Proc. of the MediaEval 2017 Workshop, Dublin, Ireland, Sept. 13-15, 2017* (2017).

[2] Ben Hoyle, Markus Michael Rau, Roman Zitlau, Stella Seitz, and Jochen Weller. 2015. Feature importance for machine learning redshifts applied to SDSS galaxies. *Monthly Notices of the Royal Astronomical Society* 449, 2 (2015), 1275–1283.

[3] Guillaume Lemaitre, Fernando Nogueira, and Christos K Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5.

[4] Gilles Louppe. 2014. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502* (2014).

[5] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. 2013. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*. 431–439.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.