DUPLICATE ANALYSIS OF ACOUSTICBRAINZ SUBMISSIONS 2018

Venkatesh Shenoy Kadandale
venkatesh.shenoykadandale01@estudiant.upf.edu

ABSTRACT

There are many duplicate submissions in AcousticBrainz, files that represent the same music recording (and have the same MBID) but were submitted by many people. These files could be from different sources (e.g. different CDs issues, a remastered CD, a recorded vinyl, a radio edit) and could be encoded using different formats (MP3, Flac, AAC). A dataset of only songs with more than one submission has been made available here. I would like to present an approach that will nominate records for ground truth data and use this ground truth to identify the most likely duplicate entries, inconclusive entries and the non-duplicate entries in this dataset. However, even in case of inconclusive entries, along with the identified duplicate and non-duplicate entries, my procedure assigns probability values for each entry which is indicative of the likelihood of its' duplicity.

1. INTRODUCTION

The musical content in the on-line media is increasing drastically in the form of music albums, sound track, singles, jingles, karaoke, background music score etc. For managing this data, it is necessary to identify and classify the audio content. This can be done by assigning a unique identifier to each of the musical track along with some descriptors which can be used for identification and classification of all the music in the digital world. AcousticBrainz project exactly does that. It crowd sources acoustic information for all music in the world and makes it available to the public. While handling crowd sourced large scale data, it is inevitable to ignore the possibility of duplicate submissions. This task aims to automatically identify the duplicate submissions. The section 2 covers the methodology involved. This is followed by results in section 3, conclusion in section 4 and future work in section 5.

2. METHODOLOGY

This is a classification task in which the dataset does not come along with the ground truth. In these kind of circumstances, it is required to observe the data distribution closely, make assumptions and then nominate certain records from the dataset for the ground truth. I believe that the way the ground truth is estimated is a critical step in this task. It does not make sense to arbitrarily take the mean of all the rows in dataset for each MBID and consider them as ground data as suggested here. Such arbitrary operations without giving regard to frequency distributions can completely distort the ground truth from reality. Especially, when the noisy data has features which differ significantly in magnitude with respect to the nominal ground truth, the inaccurate ground truth can render the entire analysis ineffective. So, it is worthwhile to carefully observe the dataset and feature distribution to estimate ground truth.

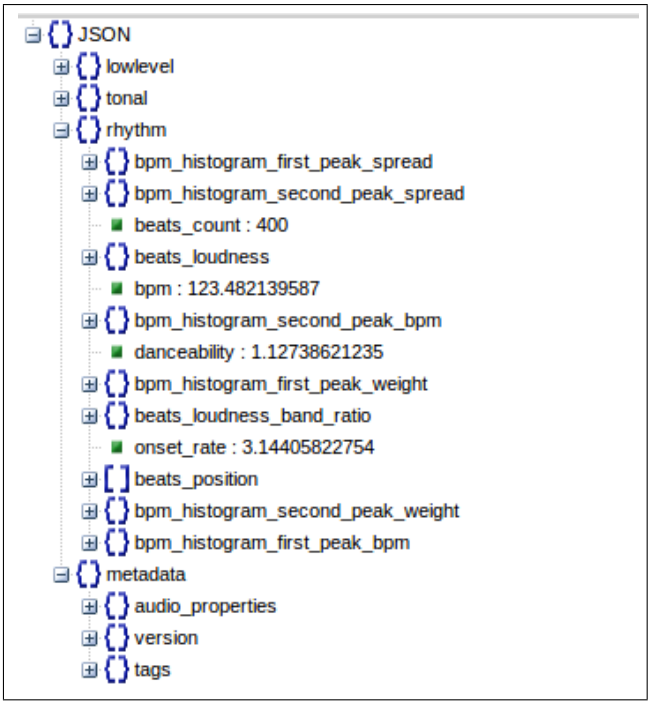

Figure 1. A sample JSON from the dataset.

2.1 Observations on Dataset

The dataset has been made available in the form of an archive of JSON files. There are totally 42990 JSON files spanning over the repetitions of 1026 unique MBIDs. Each JSON file represents a song and contains information like metadata, acoustic and musical features as seen in Figure 1. These JSON files contain a lot of information and most of them do not seem to be relevant for this task. I find it logical to consider the feature set {'id', 'folder', 'filename' ,'title', 'length', 'bpm', 'loudness', 'onset_rate', 'key_key', 'key_scale', 'replay_gain', 'tuning_frequency'}

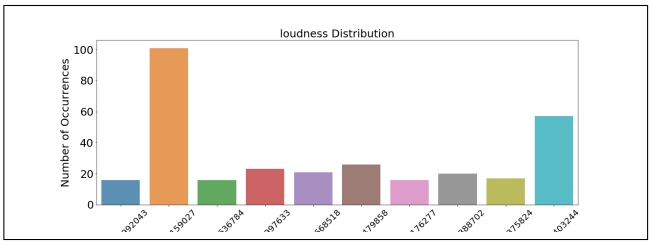

Figure 2. Loudness Frequency Distribution (Top 10)

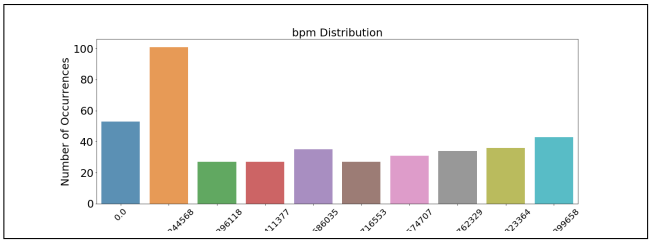

Figure 3. BPM Frequency Distribution (Top 10)

to begin with. Ideally, for duplicate submissions these features should not vary much. To know more about the relevance of these features for this particular task, I studied their frequency distributions.

2.2 Visualization of Feature Distribution

In this section, the frequency distribution of each feature is investigated. Top 10 elements of this distribution is plotted for each feature. These plots give insight regarding the relevance of the features for the task.

In Figure 6, I noticed that key_scale is binary and will not be much helpful. The statistical moments, especially the variance/(second moment/) and kurtosis/(fourth moment/), for the distributions of {'id', 'length', 'bpm', 'key_key'} are very close to each other than that of 'title' and 'tuning_frequency' as it appears from Figure 2, Figure 9, Figure 4, Figure 5,Figure 6, Figure 7 and Figure 8]. Hence, I would like to drop 'title' and 'tuning_frequency'. At this stage, I am with this features set : {'id', 'length', 'bpm', 'key_key', 'replay_gain', 'onset_rate', 'loudness'}.

2.3 Ground Truth Estimation

This step is based on the assumption that if multiple records with same MBID have the same values for critical descriptors from the feature set, then they are duplicates.

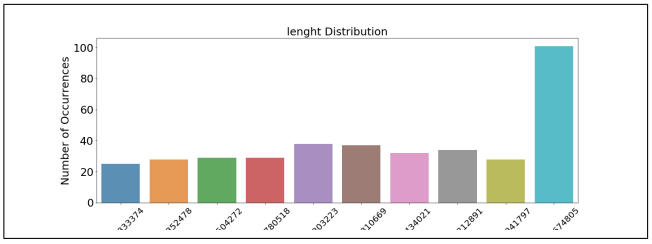

Figure 4. Length Frequency Distribution (Top 10)

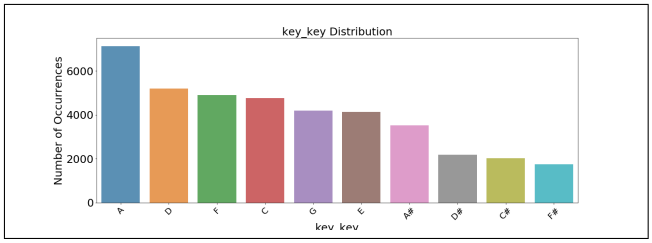

Figure 5. Key Frequency Distribution (Top 10)

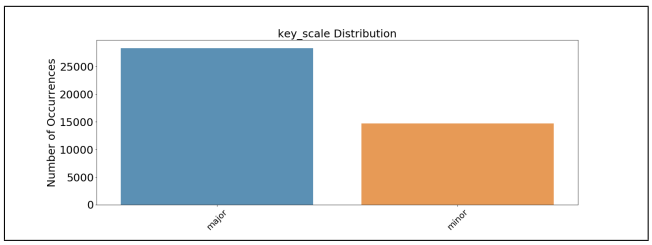

Figure 6. Key Scale Frequency Distribution (Top 10)

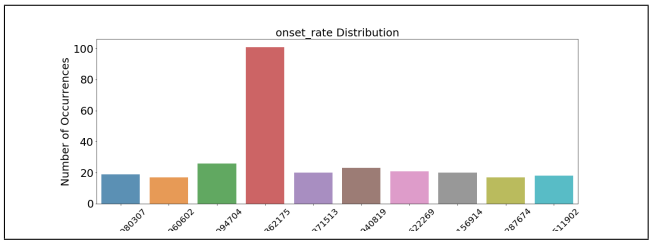

Figure 7. Onset Rate Frequency Distribution (Top 10)

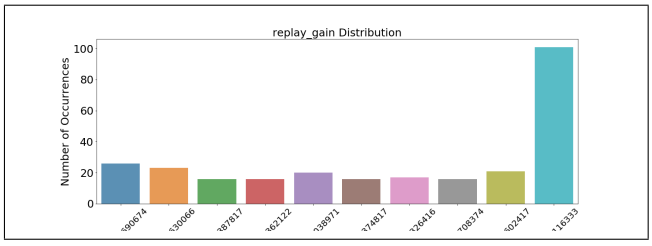

Figure 8. Replay Gain Frequency Distribution (Top 10)

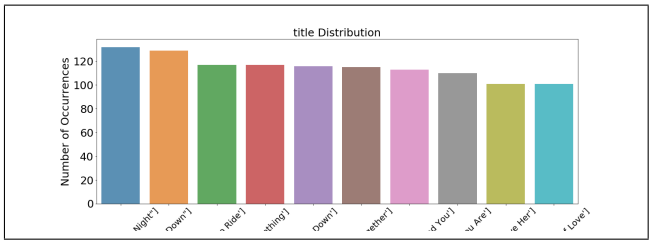

Figure 9. Title Frequency Distribution (Top 10)

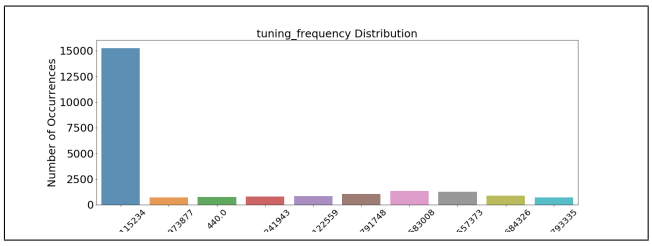

Figure 10. Tuning Frequency Frequency Distribution (Top 10)

These descriptor values can then be used as the ground truth for that particular MBID. We then specify tolerance for deviation from these critical descriptor feature vector to decide whether the entry is duplicate or not. The critical descriptors are determined as follows:

- The critical descriptors need to be such that the frequency distribution of the MBIDs grouped by this set of critical descriptors will have the maximum spread. In other words, the desired frequency distribution should be having high variance and low kurtosis. This is based on the assumption that the each MBID has a uniform tendency to get more duplicates. The entries in the top of the frequency distribution have very high number of duplicates. My assumption states that the other MBID entries have tendency to have duplicity up to the top MBID entry in frequency distribution.
- However, there is a constraint.The confidence of the analysis will be depending on the number of descriptors that are shortlisted from the features set. However, with more descriptors, the kurtosis increases and this is not desirable.

In this way, there is a need to find the optimum trade-off between the number of shortlisted descriptors and kurtosis of the frequency distribution of the presumed duplicates. In this task, this optimization step is done manually and it can be automated. The optimization resulted in this shortlisted feature set: {'id', 'length', 'bpm', 'key_key'}.

3. EVALUATION

Firstly, the tolerance parameters for deviation in our shortlisted features set need to be configured. The parameters bpm and length are numerical and hence have numerical tolerance bands. However, key is non-numerical. Any change is key with respect to the ground truth has been considered as a deviation beyond tolerance. The deviation is computed with respect to the ground truth that is very specific to the MBID. The entries which are beyond-the-tolerance-limit in all the parameters of our shortlisted features set are very highly likely to be non-duplicates than others. For all other conditions like beyond-the-tolerance deviation in one of the parameters or two of them, the duplicity can not be decisively declared. However, two submissions with same MBIDs are more likely to be non-duplicates when they have two beyond-the-tolerance-limit parameters than one. Using this logic, I computed the probabilities of entry being non-duplicate.

For example, consider ground truth G and a data point X of n dimensional data point.

$$G : \{G_1, G_2, G_3, ..., G_n\}$$

$$X : \{X_1, X_2, X_3, ..., X_n\}$$

Let $T : \{T_1, T_2, T_3, ..., T_n\}$ be the tolerance threshold for the features. Let $S : \{S_1, S_2, S_3, ..., S_n\}$ be a set of 0s or 1s values such that $S_i = (abs(G_i - X_i)) > T_i$

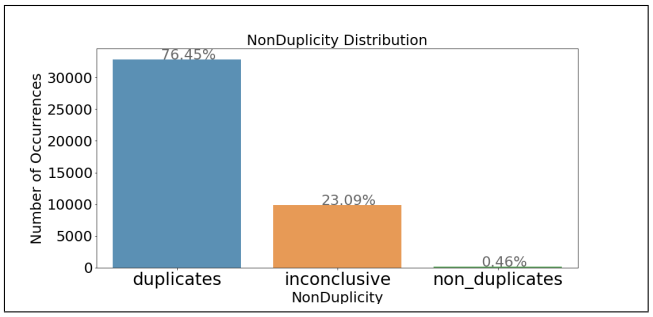

Figure 11. Duplicity Predictions

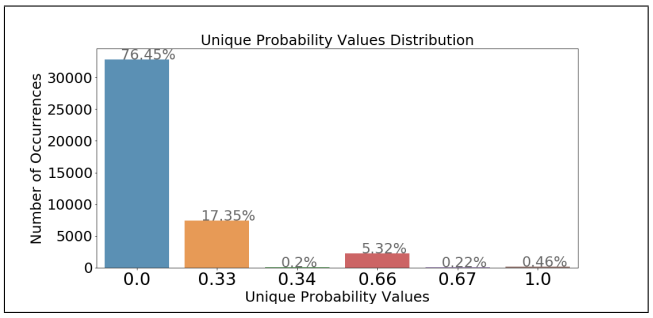

Figure 12. Unique Probability Frequency Distribution

Let $P : \{P_1, P_2, P_3, ..., P_n\}$ be the priority values with each value $\in (0, 1)$ such that $\sum_{i=1}^{n} P_i = 1$

We define a score function

$$Score(P, S) = \sum_{i=1}^{n} P_i \cdot S_i$$

Higher the score, higher the probability that the entry is not a duplicate. The maximum possible score is 1(non-duplicate) and the minimum possible score is 0(duplicate).

4. RESULTS

The Figure 11 shows the counts for each of the labels: {duplicates, inconclusive and non_duplicates} predicted for the dataset. We notice that 76.45 % of the entries are duplicate submissions, 0.46 % of the entries are non-duplicate submissions and the rest 23.09 % of the predictions for submissions are inconclusive for the threshold parameters of $bpm\_threshold = 5$ and $length\_threshold = 30$ seconds and feature set {bpm, length, key_key}. The priority values for the features are set as {'bpm' : 0.33,'length' : 0.34,'key_key' : 0.33} The computed probabilities are dumped as a separate column along with the data into a CSV file. A higher probability score indicates that the entry is non-duplicate. These results will have to be validated by humans till we find the best feature set for arriving at truthful results. The Figure 12 shows the distribution of unique probabilities for non-duplicity.

5. INSTRUCTIONS TO RUN THE NOTEBOOK

All the requirements to run the python Notebook are mentioned within the Notebook itself. Nevertheless, I will give a brief overview of the requirements below.

- The Notebook is self-sufficient in the sense that the user can start with data download and end with the reproducible results without additional support.
- The data download is taken care of and can be skipped. The feature extraction is taken care of and can be skipped.
- All the paths are provided in relative manner. The results are saved to a CSV file. The plots are saved to the plots folder.
- The Notebook makes use of these external '.py' files : 'confirm_prompt.py' for brevity of the code inside the Notebook. The prompts are used to get user input. Through this the user can make choices which can fasten the procedure.