Duplicate Analysis of AcousticBrainz submissions

Joaquin Jimenez Sauma

This report details the process, thoughts and solution developed to analyze duplicate entries in the AcousticBrainz dataset provided in class. At the end of the documents I present instructions on how to run the notebook developed for this task.

The final result of this process is a list of the entries that are considered to be mislabeled.

Loading dataset

The first approach to this problem was to look at the data, for this an online json visualizer was used.

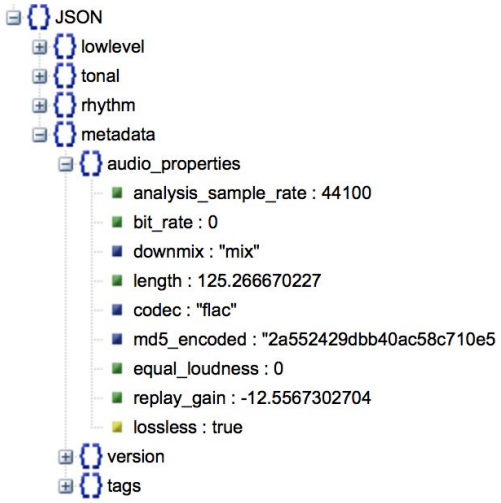

Figure 1. Structure of the dataset.

After looking at the data and information in the files, the first step was to find a way to load everything on a table in a database management system. For this tried to use the recommended *json_to_csv* process. When testing I found that this solution was not the right for this task, because some of the fields have arrays with different number of elements, and to load all of the data into a relational database was going to be a data architecture expensive task.

Then I decided to follow the recommendations on the instructions for this assignment, this means loading just the data I am analyzing:

- length
- bpm
- average loudness
- onset rate
- beat positions
- chords histogram
- hpcp mean
- key_key and key_scale
- replaygain
- tuning frequency

And load them to a datagrid structure provided by the use of Pandas, which is a library that handles data in a way similar to a database, but with the flexibility of the Python environment. This set of fields was the starting point and if time allows, of further work is needed, I will import more information. All data was loaded using *json* library.

	Unnamed: 0	id	folder	filename	title	lenght	bpm	loudness	onset_rate	key_key	key_scale	rep
0	0	[u'619ee3de-edef-4370-9373-02d90b1ccbb7']	61	619ee3de-edef-4370-9373-02d90b1ccbb7-25.json	[u'Never Tear Us Apart']	184.842453	96.367264	0.720786	2.834669	F	major	-
1	1	[u'61501d89-f666-4281-a0f8-5eaa01b5173b']	61	61501d89-f666-4281-a0f8-5eaa01b5173b-3.json	[u'School Days']	162.468567	131.371063	0.958488	4.609752	G	major	-1
2	2	[u'610c0012-6eb4-42a0-b759-3a2532ce0f15']	61	610c0012-6eb4-42a0-b759-3a2532ce0f15-23.json	[u'The Tourist']	324.597565	75.893387	0.842272	2.378202	E	major	-1
3	3	[u'619f60fa-b680-4735-a635-fc0f03715227']	61	619f60fa-b680-4735-a635-fc0f03715227-0.json	[u'What Goes On']	168.573334	97.158920	0.905077	5.457075	A	minor	-
4	4	[u'612eacef-bc8a-4797-aa72-5f811e0ffda8']	61	612eacef-bc8a-4797-aa72-5f811e0ffda8-17.json	[u'Every Little Thing She Does Is Magic']	260.623688	163.864441	0.871976	4.653858	D	minor	-
5	5	[u'61501d89-f666-4281-a0f8-	61	61501d89-f666-4281-a0f8-	[u'School	163.369707	131.294312	0.944969	4.547480	G	major	1

Table 1. Part of the loaded dataset displayed from a datagrid structure in Pandas.

Even when I load all the fields mentioned before, some of the multilevel data seems a little complex to start with. I am sure this data will provide more information and without a doubt it will give us more detail for comparison, I want to keep it simple and have results to start the analysis.

Analyzing the dataset

Having the selected data on a dataset makes things easier since all we need to do now is the actual analysis. In the dataframe, each row corresponds to a song. The approach here was to read the data and create a set of thresholds to help discriminate records. For evaluating the thresholds, new columns were added to the dataset. This columns are the result of comparing averages, medians and standard deviations for values in each group for entries for the same song (defined by the id field).

The fields added along with the formulas to define its data are:

```
# Label files where length is between 30 seconds range from mean
dfv['Length_Ok'] = np.where(abs(dfv.lenght - dfv.lenght.median()) < 30, True, False)
dfv['Lengh_Mean_Dist'] = abs(dfv.lenght - dfv.lenght.median()) # Computing distance
from lenght Mean

# Label files where bpm is in a 5 beat range from median bpm
dfv['BPM_Ok'] = np.where(abs(dfv.bpm - dfv.bpm.median()) < 5, True, False)
dfv['BPM_Mean_Dist'] = abs(dfv.bpm - dfv.bpm.median()) #Computing distance from bpm
mean

# Label files where loudness is in the range of first stanard deviation
dfv['Loudness_Ok'] = np.where((dfv.loudness < abs(dfv.loudness.std() - dfv.loudness)),
True, False)
dfv['Loudness_1st_STD_Dist'] = abs(dfv.loudness.std() - dfv.loudness)

# Label files where key_key is not the mode
dfv['Key_Ok'] = np.where(dfv.key_key == dfv.key_key.mode()[0], True, False)
```

It is important to mention that the datagrid is grouped by song or id, and this calculations are applied to a set of rows that belong to the same id. For example, median of length means the median length of all songs with the same id, not for the whole datagrid.

In Table 2 I show part of the table that contains the result of this calculations.

tuning_frequency	Length_Ok	Lengh_Mean_Dist	BPM_Ok	BPM_Mean_Dist	Loudness_Ok	Loudness_1st_STD_Dist	Key_O
438.224518	False	43.730881	False	28.792885	False	0.506795	Fals
444.085327	False	66.104767	False	6.210915	False	0.744497	Fals
438.984558	False	96.024231	False	49.266762	False	0.628280	Fals

Table 2. Calculated fields.

The fields are showing the data I want to discard. *Lenght_Ok* shows when an entry has a length that is within 30 seconds difference from the mean of all entries for this id. *BPM_Ok*, does the same, showing when an entry is within a 5 beat range from the mean of this Id. *Loudness_Ok* shows if this entry is in the first standard deviation of all the values for this Id. *Key_Ok*, tells if the entry provided key value is the same as the mode for the entries with this Id.

To select the entries that are considered mislabeled I just selected the ones that have all OK fields as 'False'. This results are saved to a file.

Analysis of the results and further work

When comparing some of the results with the original data, I found that this simple process creates very accurate results, while just scratching the surface. The entries selected as mislabeled are really different from most of the data, even when we don't know exactly the audio recording used to create each entry. In this case, the analysis of fields like beat positions, chords histograms and beat onsets will provide more information. For this analysis, priority was given to the fields of length, BPM, Key and Loudness.

A BPM of half or double the real value for a song is a problem that makes some entries to be labeled as mislabeled. Since this values cause erroneous average or median computation, the whole set of entries for the same Id can be discarded. A quick possible solution could be to set thresholds for allowed values of BPM field or create a process that can compare values that may be double or half, in order to standardize this values.

Some tracks were not recorded with a constant tempo, and the reported BPM doesn't represent the whole recording. For this and the double/half BPM issue, we would need to use the beat positions information to make sure we are comparing the same information. A more complex algorithm would be needed to develop this solution.

I find hard to think that a definitive analysis can be performed without ground truth, but since it is not available and maybe it won't, this exercise provides a good insight of the dataset provided.

How to run the notebook

The notebook uses the folder 'ab-duplicates1000-2016-03-02' as input (created when unzipping the dataset file) and outputs the mislabeled entries inside this folder, on a file specified in the variable `resultsFile`.

It is necessary to update variables in the first cell so the process can find the source data. After this, the notebook can be run entirely.

The process of loading data from json files is lengthy because the amount of information processed. In order to avoid reading the data each time the process is run, the processed information is saved to a file specified by the `preProcessedFile` variable, and it is saved in the same folder as the source data. You can leave this variable with its current value. This file is provided in the zipped file for this assignment and if you want to avoid the loading process, you can copy the file '`duplicateanalysis.csv`' to the folder containing the dataset and run the process from cell 4. The results of this analysis are saved to the same folder.