AMP LAB TASK 2
Agreeement between AcousticBrainz vs MSD data

Marc Siquier Peñafort

February 13, 2017

All the code for this task can be found in: https://github.com/Marcsiq2/aspma-lab2

1 Exploring Data

For this task we'll have four different datasets: **AB_LOWLEVEL** that contains all low_level features from *AcousticBrainz* (AB) indexed by `mbid` (Music Brainz ID). **AB_BEATS** contains all the beats positions from AB, also indexed by `mbid`. Then we have **MSD_SUMMARY** that contains all low_level features and beats positions and beats confidence from *Milliong Song Database* (MSD) indexed by `msd_id` (Million Song Database ID). **MSD_BEATS** contains all the beats positions and beat confidence from msd, also indexed by `msd_id`. Finally in order to map `mbid` and `msd_id` we have a dataset **MBID_MSD** that contains the mapping and also the song name.

First step in order to be able to evaluate all the data is to explore it and pre-process it, so, first of all, I checked the number of instances in each database and figured out that there are quite different.

Database	Instances
AB_LOWLEVEL	2352902
AB_BEATS	778211
MSD_SUMMARY	237962
MBID_MSD	282046

Table 1: Number of Instances

From this table 1 we see that we first need to obtain the common instances from all dataset: Instances that are both in AB and in MSD and that the mapping is in MBID_MSD dataset. Moreover I will do a duration check for both AB and MSD instances so we are sure that the song is the same. For this duration check we will set a duration precision window percentage that will allow some precision error to the metric.

Duration pw	Common Instances
0.50	268749
0.40	263293
0.20	244758
0.10	227137
0.05	**212147**
0.01	161069

Table 2: Common Instances

As we can see from table 2 in some cases, the number of common instances is larger than the number of instances in the MSD_SUMMARY database. This is because some AB instances have more than one corresponding instance in MSD_SUMMARY dataset. From all this possible common instances we will take for the next experiments a duration precision window of 0.05 (5% duration difference) with a total common instances of 212147.

2 AB bpm vs MSD tempo

In order to evaluate AB bpm vs MSD tempo we will consider two evaluation metrics:

2.1 Accuracy 1

The percentage of AB bpm estimates within 4% (the precision window) of the MSD tempo, or vice versa. In order to compute this, I implemented a simple evaluation function as it can be seen in the following pseudo-code:

function $evaluate\_bpm(bpm_1, bpm_2, p_w)$
  $diff := |bpm_1 - bpm_2|$
  if $diff \leq (bpm_2 * p_w)$ **then return** True
  elsereturn False
  end if
end function

Using this simple metric and different precision windows I came up with a BPM accuracy 1 results table 3 comparing both MSD as ground truth and AB as ground truth. From this table we can see that with a precision window of 4% we obtain an accuracy of 64% for both ground truths. As we allow more error in the estimation we can see as the accuracy also raises.

	MSD as GT		**AB as GT**	
Precision window	Correct instances	Accuracy	Correct instances	Accuracy
0.01	114814	0.5412	114803	0.5411
0.04	136550	0.6437	136571	0.6438
0.10	144487	0.6811	144517	0.6812
0.25	153613	0.7241	154075	0.7263
0.50	180196	0.8494	182860	0.8619

Table 3: BPM Accuracy 1

2.2 Accuracy 2

The percentage of AB bpm estimates within 4% of either the MSD tempo, or half, double, three times or one third of the MSD tempo. In order to compute this accuracy we will feed into accuracy 1 *evaluate_ bpm* metric algorithm the MSD tempo value and the AB bpm itself, multiplied by two, by three and divided by two and by three.

function $evaluate\_bpm2(bpm_1, bpm_2, p_w)$
  if $evaluate\_bpm(bpm_1, bpm_2, p_w)$ **then return** True
  else if $evaluate\_bpm(bpm_1 * 2, bpm_2, p_w)$ **then return** True
  else if $evaluate\_bpm(bpm_1 * 3, bpm_2, p_w)$ **then return** True
  else if $evaluate\_bpm(bpm_1/2, bpm_2, p_w)$ **then return** True
  else if $evaluate\_bpm(bpm_1/3, bpm_2, p_w)$ **then return** True
  elsereturn False
  end if
end function

Using this enhanced bpm metric and different precision windows we obtain the BPM accuracy 2 table 4 comparing also both MSD as ground truth and AB as ground truth. From this table we can see that with a precision window of 4% we obtain an accuracy of 77% for both ground truths. This value is 12% higher than with accuracy 1 metric so we see that there are a lot of subdivision errors when computing bpm.

	MSD as GT		**AB as GT**	
Precision window	Correct instances	Accuracy	Correct instances	Accuracy
0.01	134216	0.6327	134193	0.6325
0.04	162810	0.7674	162831	0.7675
0.10	174284	0.8215	174252	0.8214
0.25	193427	0.9118	192980	0.9097
0.50	211680	0.9978	211760	0.9982

Table 4: BPM Accuracy 2

3 AB key_key/key_scale vs key/mode

In order to evaluate key results from AB and MSD a preprocessing stage was necessary for MSD data in order to format it in the correct way for the evaluation part. For evaluation I used the function *key.weighted_score* from **mir_eval** Python library [1] which computes a heuristic score weighted according to the relationship of the reference and estimated key. I also compared data in both ways, MSD as ground truth and AB as ground truth and as we can see in the results (table 5. Depending on the *key_confidence* value I discard all key values from MSD dataset that are below the threshold.

Key confidence	Total instances	MSD as GT	AB as GT
0.9	1239	0.7881	0.7490
0.7	19128	0.6870	0.6870
0.6	45100	0.6248	0.5950
0.5	79185	0.5617	0.5373
0.3	140231	0.4796	0.4635
0.0	212147	0.4103	0.4018

Table 5: Key evaluation

4 AB beats_position vs MSD beats_start

For this last subtask we will evaluate how both datasets (AB and MSD) correlate estimating beats positions. From MSD data we have also the `beat_confidence` for each beat so we can filter this beats in order to consider different confidence levels in evaluation. I also did a common preprocessing stage which consists in removing the beats present in the first 5 seconds.

Mir_eval library has been also used in order to score beats positions. In particular, I used `f_measure` function present in the beat module from the library. The results in table 6 show the average f_measure for all instances depending on the confidence threshold and also considering GT and AB as ground truth.

NOTE: This task is very cpu and memory consuming, so we need to restart python notebook before running it. Wall time in the table 6 shows the real time that elapses from start to end of the task being performed. As we increase MSD beats confidence, we need to compare more beats so wall time will also increase.

MSD Beats Confidence	MSD as GT F-score	AB as GT F-score	Wall time[2]
0.9	0.0181	0.0181	4min 22s
0.8	0.0436	0.0436	4min 51s
0.7	0.0805	0.0805	5min 58s
0.6	0.1729	0.1729	9min 48s
0.3	0.2465	0.2465	15min 32s

Table 6: Beats Positions

[1] https://craffel.github.io/mir_eval/

[2] Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz, 4096KB CPU cache, 16GB RAM