**Aggregating algorithm**

As per the project brief, we have provided a python file (aggregate\_algorithm.py) which implements the algorithm laid out in “A generalized approach for producing, quantifying, and validating citizen science data from wildlife images” (Swanson et al.) in order to create aggregated classifications for the photos taken as part of the MammalWeb project.

Aggregate\_algorithm uses the pymysql library to handle the connection with the SQL database in which all the photo and classification data is stored.

The function provided should be run server side whenever a new classification is made for a photo by a ‘spotter’ on the mammalweb website.

**aggregate\_algorithm(photo\_id,connection,preBlank(=none),preflag(=none))**

connection is a pymysql connection object which gives access to a database. The algorithm finds all individual classifications for the photo with ‘photo\_id’ and combines them as per Swanson et al. into a row in a new database table, ‘aggregate’. As part of the implementation, we have to find which options represent the ‘blank’ options and which are ‘flags’, these values can be passed into the function if the algorithm is being run on a set of photos at once to save a small amount of calculation time. The parameters are ‘none’ by default and these values are calculated at runtime if not passed in.

**Outputs**

The function outputs a 9-tuple containing all of the aggregate attributes and metrics along with the photo\_id. These are all the attributes stored in the aggregate table.

The evenness, (fraction) support and (fraction) blanks attributes of the output records measure the likelihood that the species given by the aggregate is correct. The age and gender are more simply the mode of the individual classifications.

The attribute ‘flag’ is a marker to show what state the aggregate for that photo has reached. These are laid out in Swanson et al. and have been added to the options table with their struc attribute as ‘flag’. Photos with a flag of ‘complete’ or ‘consensus’ should be removed from the pool of photos to be classified and the aggregate considered correct. Photos with a flag of ‘blank’ should also be removed as they almost definitely do not contain any animal. Photos with a flag of ‘incomplete’ should remain in the rotation as they require more classifications to be sure of the aggregates accuracy. Finally, if any flags are set to -1 an error has occurred and needs to be corrected. Flag values are contained in the options table with a struc attribute of ‘flag’.

To make use of aggregate\_classification, aggregateAll, aggregateOne and aggregateRange functions are included. These run aggregate\_classification and store the results in the aggregate table on every photo, a single photo or a selection of photos respectively.

Pymysql is very good at running multiple INSERT queries thanks to the executemany function which concatenates all the values to insert into a single query, but is not very good for running a large number of UPDATE or DELETE queries as these have to be done individually. For this reason, when running aggregateAll, the entire aggregate table is truncated just before reinserting all of the values. When running aggregteRange, if the range is more than around 200 entries the delete operations will begin to slow down the process. As the algorithm is designed to be run on individual photos whenever there is a new classification for it, this shouldn’t be a problem.

Also included is a test function, checkAgainsGoldStandard to compare aggregate classifications with a gold standard data set that was provided.

This implementation uses the pymysql library available at <https://github.com/PyMySQL/PyMySQL> under a free use and distribution license.

Results

On an intel i5-4310M 2.7GHz, running aggregateAll with a 42068 photo database takes around 19 seconds (0.00045165 sec/photo on average). Running aggregateOne takes an average of 0.1124 seconds/photo. This is significantly longer due to the insert and delete/truncate queries taking around the same regardless of the amount of entries being inserted/deleted.

In total, 96% of the aggregate species are the same as the gold standard data.

When considering only the complete and consensus flagged aggregates, agreement is just over 99%.

The species and age fields are simply the mode of all the classifications but both agree with the gold standard on around 75% of photos. This is likely because