Scene Classification task: specification

DATA

The data consists of 30-second audio files (WAV, stereo, 44.1 kHz, 16-bit), recorded using binaural headphones in locations around London at various times in 2012, by three different people. Locations were selected to represent instances of the following 10 classes:

bus
busystreet
office
openairmarket
park
quietstreet
restaurant
supermarket
tube
tubestation

The unreleased train/test dataset consists of 10 recordings of each class, making 100 recordings total. This is similar to the publicly-released development dataset.

EVALUATION

Participating algorithms will be evaluated with 5-fold stratified cross validation.

The raw classification (identification) accuracy, standard deviation and a confusion matrix for each algorithm will be computed.

In addition computation times of each participating algorithm will be measured.

INPUT AND OUTPUT FILE FORMATS

The audio files to be used in these tasks will be specified in simple ASCII list files. The formats for the list files are specified below:

Training list file

The list file passed for model training will be a simple ASCII list file. This file will contain one path per line, followed by a tab character and the class label, again with no header line. I.e.

```
<example path and filename>\t<class label>
```

e.g.

```
/path/to/track1.wav tubestation
/path/to/track2.wav park
...
```

Test (classification) list file

The list file passed for testing classification will be a simple ASCII list file with one path per line with no header line, and no class label. I.e.

```
<example path and filename>
```

E.g.

```
/path/to/track1.wav
/path/to/track2.wav
...
```

Classification output file

Participating algorithms should produce a simple ASCII list file identical in format to the Training list file. This file will contain one path per line, followed by a tab character and the scene label, again with no header line. I.e.

```
<example path and filename>\t<class label>
```

E.g.

```
/path/to/track1.wav tubestation
/path/to/track2.wav park
...
```

There should be no additional tab characters anywhere, and there should be no whitespace added after the label, just the newline.

SUBMISSION CALLING FORMATS

Executables must accept command-line parameters which specify:
- A path to a training list file
- A path to a test list file
- A path to specify where the classification output file will be written
- A path to a scratch folder which the executable can optionally use to write temporary data

Executables must NOT write data anywhere except the classification output file and the scratch folder.

A typical entry-point for your submission could be for us to run a command such as one of these:

```
   TrainAndClassify.sh /path/to/scratch/folder
/path/to/trainListFile.txt /path/to/testListFile.txt
/path/to/outputListFile.txt

   python smacpy.py -q --trainlist /path/to/trainListFile.txt
--testlist /path/to/testListFile.txt --outlist
/path/to/outputListFile.txt
```

PACKAGING SUBMISSIONS

For Python/R/C/C++/etc submissions, please ensure that the submission can run on the Linux disk image we provide, WITHOUT any additional configuration. You may have modified the virtual machine after downloading it, but we will not be using your modified disk image - we will be running your submission on the standard disk image. This means:
- if you have used additional Python/R script libraries, they must be included in your submission

bundle, and your script should be able to use them without installing them systemwide.
- if you have used any additional C/C++ libraries, they must be statically-linked to your executable.

For Matlab submissions, ensure that the submission can run with the toolboxes and system that the organisers have specified. If you need any particular toolboxes or configuration please contact the organisers as soon as you can. Please aim to make MATLAB submissions compatible across multiple OS (usual problems exist in the file/path separators). All Matlab submissions should be written in the form of a function, e.g. eventdetection(input,output); which can allow calling the script from the command line very easily.

Please provide some console output, which can provide a sanity check to the challenge team when running the code. This can be of the form of simply writing out a line corresponding to different stages of your algorithm.

All submissions should include a README file including the following the information:

- Command line calling format for all executables including examples
- Number of threads/cores used or whether this should be specified on the command line
- Expected memory footprint
- Expected runtime
- Approximately how much scratch disk space will the submission need to store any feature/cache files?
- Any special notice regarding to running your algorithm

Note that the information that you place in the README file is extremely important in ensuring that your submission is evaluated properly.

Time and hardware limits

Due to the potentially high resource requirements across all participants, hard limits on the runtime of submissions will be imposed. A hard limit of 48 hours will be imposed for each submission.