Annotation scripts README

This README describes the various scripts available for doing manual segmentation of media files, for annotation or other purposes, and converting from-to the file formats of several related tools.

The scripts are either in *python2* or *perl*, but interpreters for these should be readily available.

Please send any questions/suggestions to antonio.macias.ojeda@aalto.fi

mseg.py

Script to help perform manual segmentation of a media file, it can be any media file type supported by mplayer. It's only dependency is a Python-mplayer wrapper that can be installed locally by executing:

```
$ easy_install --user mplayer.py
```

After that executing it is just:

```
$ ./mseg.py /path/to/mediafile -o outputfile
```

The output file is optional. It also supports the invocation:

```
$ ./mseg.py /path/to/mediafile -o outputfile -i inputfile
```

To continue a previously saved segmentation session. Once in the program, the controls are:

- Quit: *esc* or *q*
- Pause: *p*
- Mark position: *space*
- Manually edit mark: *e*
- Add manual mark: *a*
- Remove mark: *r*
- Faster speed: *Up*
- Slower speed: *Down*
- Rewind: *Left*
- Fast Forward: *Right*
- Scroll down marks: *pgDwn*
- Scroll up marks: *pgUp*

The media file starts as paused, so to start reproduction just hit the *p* key.

mseg2elan.py

Script to convert from mseg output to Elan file format.

Usage:

```
$ ./mseg2elan.py msoutputfile -o outputfile
```

If *outputfile* is not specified, the output will be sent to the stdout. Once in Elan, segments can be easily fine tuned by changing to the segmentation mode, in Options->Segmentation Mode.

aku2elan.py

Script to convert from AKU recipes to Elan file format.

Usage:

```
$ ./aku2elan.py recipe -o outputfile
```

If *outputfile* is not specified, the output will be sent to the stdout. Once in Elan, segments can be easily fine tuned by changing to the segmentation mode, in Options->Segmentation Mode.

elan2aku.py

Script to convert from Elan file format to AKU recipes.

Usage:

```
$ ./elan2aku.py elanoutputfile -o akurecipe
```

If *akurecipe* is not specified, the output will be sent to the stdout.

mseg*to*textgrid.pl

Script to convert from mseg output to praat file format.

Usage:

```
$ perl mseg_to_textgrid.pl msfile > outputfile
```

If *outputfile* is not specified, the output will be sent to the stdout.

voice-detection.py

Creates an AKU recipe from the *classify_speecon* output (.exp files).

For full help, use:

$ ./voice-detection.py -h

vad-performance.py

Rates the performance of a Voice Activity Detection recipe in AKU format, such as those created with *voice-detection.py*. To measure the performance, another recipe with ground truth should be provided.

For full help, use:

$ ./vad-performance.py -h

spk-change-detection.py

Performs speaker turn segmentation over audio, using a distance measure such as GLR, KL2 or BIC, and sliding or growing window. It requires an input recipe file in AKU format pointing to the audio files, and preferably with turns of speech/non-speech already processed, and a features file for each wav to process, in the format outputted by the feacat program of the AKU suite.

For full help, use:

$ ./spk-change-detection.py -h

spk-change-performance.py

Rates the performance of a speaker turn segmentation recipe in AKU format, such as those created with *spk-change-detection.py*. To measure the performance, another recipe with ground truth should be provided.

For full help, use:

$ ./spk-change-performance.py -h

spk-clustering.py

Performs speaker turn clustering over audio. It requires a speaker segmentation recipe in AKU format, such as those created with *spk-change-detection.py*, and a features file for each wav file to process, in the format outputted by the feacat program of the AKU suite.

For full help, use:

$ ./spk-clustering.py -h

spk-time.py

Calculates per-speaker speaking time from a speaker-tagged recipe in AKU format.

For full help, use:

$ ./spk-time.py -h

spk-diarization.py

Performs full speaker diarization over media file. If the media is not a wav file it tries to convert it to wav using ffmpeg. It then calls *classify_speecon.pl*, *voice-detection.py*, *spk-change-detection.py* and *spk-clustering.py* in succession.

For full help, use:

```
$ ./spk-diarization.py -h
```

Notes:

- Paths for the other scripts and features must be provided.
- *classify_speecon* must be properly configured for this to work, edit it and related files to ensure that all paths are set properly or it will fail.
- Since this script is a convenient wrapper for the other scripts of the family, it doesn't have options for all the settings of the other scripts, just some defaults. If you want to tune them, edit this script directly.