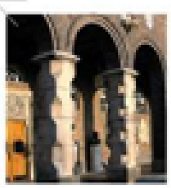
SZEGEDI TUDOMÁNYEGYETEM
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS

Detecting Laughter and Filler Events by Time Series Smoothing with Genetic Algorithms

Gábor Gosztolya
MTA-SZTE Research Group on Artificial Intelligence
and
University of Szeged,
Szeged, Hungary



Social Signal Detection

- Social Signals
  - Laughter and filler events (sounds like ``eh'', ``er'', ``um'' etc.)
  - They regulate the flow of interaction in discussions
  - Their detection has became popular recently
- Model training and evaluation
  - Models are trained and evaluated on the frame-level
  - The standard evaluation metric is Area-Under-Curve (AUC) for the output posterior scores
  - It is worth using the contextual information (i.e. the neighbouring frames) during training and evaluation

Model Training and Evaluation

- Frame-level approach
  - 10ms frame shift
  - Classifier: GMM, ANN/DNN, Gaussian Processes…
  - Use the feature vectors of the neighbouring frames
- Local score aggregation after classifier evaluation
  - It is worth to adjust the frame-level **output scores** based on the local neighbourhood (``smoothing”)
  - Gupta et al. (2013): probabilistic time series smoothing
  - Brückner (2014): smoothing by DNN
  - Gosztolya (2015): Simple Exponential Smoothing

Output Score Aggregation

- Classifier output score aggregation
  - The optimal way of score aggregation is not clear
  - We chose the weighted form of the moving average time series filter
  - A filter takes the form $w_{-N}, \ldots, w_{-1}, w_0, w_1, \ldots, w_N$ with a length of $2N+1$
  - For the jth frame with the raw likelihood estimate a_j we simply calculate

$$a'_j = \sum_{i=-N}^{N} w_i a_{j+i}.$$

  - We use the simplification that for all $j < 1$, $a_j=a_1$; and for all $j > k$ (the length of utterance) $a_j=a_k$

The SSPNet Vocalization corpus

- Contains English spontaneous conversations over telephone
  - 2763 30-seconds long clips from 120 speakers
  - 2988 laughter and 1158 filler events
- Featured in the Interspeech Computational Paralinguistic Challenge (ComParE) in 2013
  - Standard train / dev / test division: 1583 / 500 / 680
  - 141-sized feature set per frame (MFCC, F0, zero-crossing rate, HNR, derivatives + mean/std over a 9-frames long window)
  - Metric: AUC, averaged for the two social signals
  - Baseline approach: linear SVM (Weka)

Classification Methods

- AdaBoost.MH:
  - An efficient meta-learner algorithm, training weighted sum of simple *base learners*
  - We used 8-leaved decision trees as base learners
  - Trained on 17 consecutive frame vectors for 100,000 iterations
- Deep Neural Networks (DNN):
  - ANN with several hidden layers
  - We used the rectifier activation function in the hidden layers, and the softmax function in the output
  - 5 hidden layers, each containing 256 neurons
  - Trained on 31 consecutive frame vectors

Genetic Algorithms

- We optimized the **w** weight vector by GA
- GAs are adaptive methods for optimization tasks
  - Their mechanisms and terminology are based on the genetic processes of biological organisms
  - A *population* (set) of *individuals* (numeric vectors)
  - Individuals consist of *genes* (parameters)
  - Each individual is assigned a *fitness score*
  - Individuals with higher fitness scores can ``reproduce'' by *crossover*, then *mutation* can happen
  - This is repeated for several *generations*; the individual of the last generation with the highest fitness will be the solution of the optimization task

Applying GA

- We optimized the **w** weight vectors by GA
  - Each filter was 129 frames long (64-64 on both sides)
  - Only each 8th weight was stored, the rest was linearly interpolated to reduce vector size to 17
  - Four filters overall (2 classifiers and 2 social signals)
  - We used the development set for optimization
- We used the JGAP package
  - 250-sized populations for 100 generations
  - We used averaging crossover
  - Mutation (replacing one weight with a random value) happened with a probability of 0.001
  - Before evaluation, the weight vectors were normalized to add up to one (normalization)

Results Without Filters

ML Method	Filter type	Dev. set			Test set		
		Lau.	Fil.	Avg.	Lau.	Fil.	Avg.
AdaBoost	---	94.0	94.9	94.5	91.9	87.9	89.9
DNN	---	92.9	95.5	94.2	91.3	87.9	89.6
SVM (ComParE 2013 baseline)		86.2	89.0	87.6	82.9	83.6	83.3

- The ``raw'' output scores outperform those of ComParE baseline SVM
- AdaBoost performed somewhat better than DNN
  - Probably due to instance sampling used during model training, which balanced the distribution of the three classes (laughter, filler, other)

Results of Filters

ML Method	Filter type	Dev. set			Test set		
		Lau.	Fil.	Avg.	Lau.	Fil.	Avg.
AdaBoost	---	94.0	94.9	94.5	91.9	87.9	89.9
	Random	97.7	94.2	95.9	94.6	87.5	91.0
	Constant	97.8	94.1	95.9	94.7	87.6	91.2
	GA	**98.0**	**96.4**	**97.2**	**95.0**	**89.5**	**92.2**
DNN	---	92.9	95.5	94.2	91.3	87.9	89.6
	Random	96.7	94.4	95.5	94.2	86.9	90.5
	Constant	**96.9**	94.3	95.6	**94.4**	86.9	90.7
	GA	96.7	**96.5**	**96.6**	94.3	**88.8**	**91.6**

- The GA-optimized filters significantly outperform raw scores and two basic filters of the same length

Results

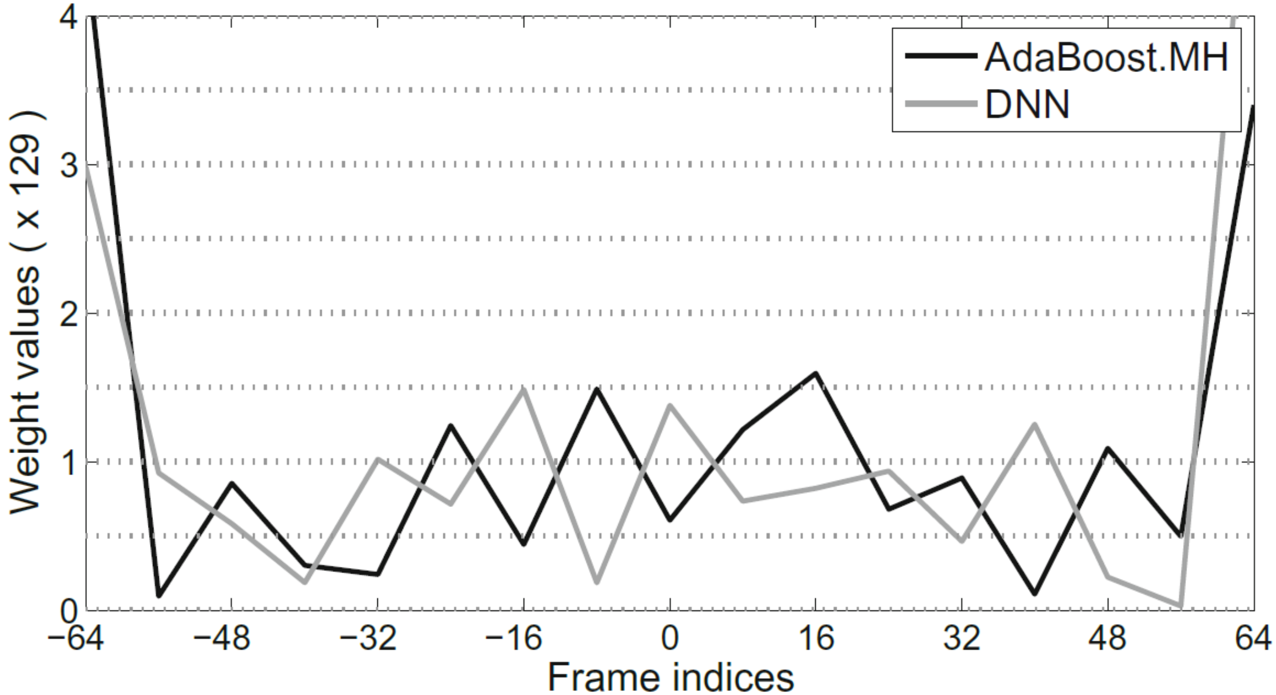
ML Method	Filter type	Dev. set			Test set		
		Lau.	Fil.	Avg.	Lau.	Fil.	Avg.
AdaBoost	---	94.0	94.9	94.5	91.9	87.9	89.9
	GA	98.0	96.4	**97.2**	95.0	89.5	**92.2**
DNN	---	92.9	95.5	94.2	91.3	87.9	89.6
	GA	96.7	96.5	**96.6**	94.3	88.8	**91.6**
DNN + Prob. TS smoothing		95.1	94.7	94.9	93.3	89.7	91.5
DNN + DNN		98.1	96.5	97.3	94.9	89.9	92.4

- The GA-optimized filters also outperform probabilistic time series smoothing (winner of ComParE 2013), although slightly lag behind DNN+DNN (which solution, by the way, did not work for us)

All Results

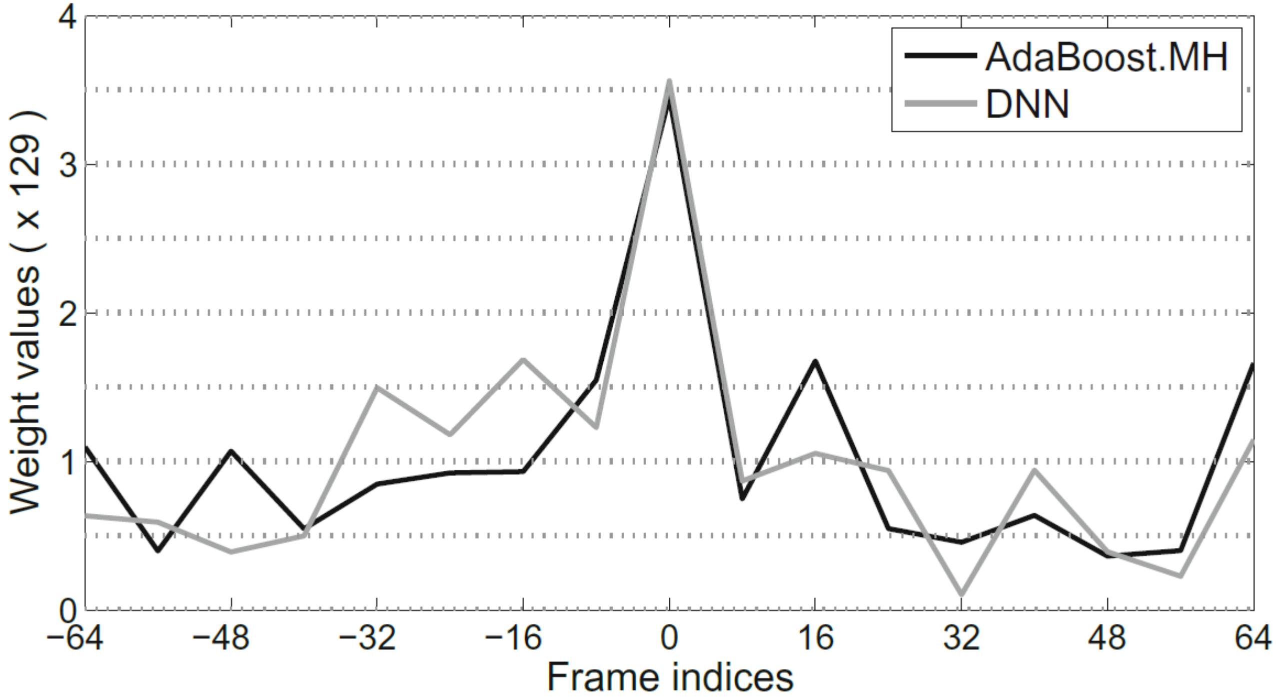
ML Method	Filter type	Dev. set			Test set		
		Lau.	**Fil.**	**Avg.**	**Lau.**	**Fil.**	**Avg.**
AdaBoost	---	94.0	94.9	94.5	91.9	87.9	89.9
	Random	97.7	94.2	95.9	94.6	87.5	91.0
	Constant	97.8	94.1	95.9	94.7	87.6	91.2
	GA	**98.0**	**96.4**	**97.2**	**95.0**	**89.5**	**92.2**
DNN	---	92.9	95.5	94.2	91.3	87.9	89.6
	Random	96.7	94.4	95.5	94.2	86.9	90.5
	Constant	**96.9**	94.3	95.6	**94.4**	86.9	90.7
	GA	96.7	**96.5**	**96.6**	94.3	**88.8**	**91.6**
DNN + Prob. TS smoothing		95.1	94.7	94.9	93.3	89.7	91.5
DNN + DNN		98.1	96.5	97.3	94.9	89.9	92.4
SVM (ComParE 2013 baseline)		86.2	89.0	87.6	82.9	83.6	83.3

Filters Found for Laughter Events



- Linear interpolation and noise is visible
- Filters found for the two classifiers are very similar
- First/last frames are very important

Filters Found for Filler Events



- The central frames are very important
- Last frame is also important; first one is only averagely

Summary

- Detecting social signals in speech is a task gaining importance lately
- After the classification and evaluation steps, it is worth adjusting the frame-level output scores
- We applied a weighted average time series smoothing filter
- The weights were set by Genetic Algorithm
- We experimented with two social signals and two machine learning methods
- The proposed method outperforms the raw scores as well as several basic and standard filters in terms of AUC