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

Gábor Gosztolya[1,2(✉)]

[1] Institute of Informatics, University of Szeged,
Dugonics tér 13, Szeged 6720, Hungary
ggabor@inf.u-szeged.hu
[2] MTA-SZTE Research Group on Artificial Intelligence,
Tisza Lajos krt. 103, Szeged 6720, Hungary

Abstract. Social signal detection, where the aim is to identify vocalizations like laughter and filler events (sounds like "eh", "er", etc.) is a popular task in the area of computational paralinguistics, a subfield of speech technology. Recent studies have shown that besides applying state-of-the-art machine learning methods, it is worth making use of the contextual information and adjusting the frame-level scores based on the local neighbourhood. In this study we apply a weighted average time series smoothing filter for laughter and filler event identification, and set the weights using genetic algorithms. Our results indicate that this is a viable way of improving the Area Under the Curve (AUC) scores: our resulting scores are much better than the accuracy of the raw likelihoods produced by both AdaBoost.MH and DNN, and we also significantly outperform standard time series filters as well.

Keywords: Social signals · Laughter and filler event detection · Time series filter · Genetic algorithms

1 Introduction

In speech technology an emerging area is paralinguistic phenomenon detection, which seeks to detect non-linguistic events (laughter, conflict, etc.) in speech. One task belonging to this area is the detection of social signals, from which, perhaps laughter and filler events (vocalizations like "eh", "er", etc.) are the most important. Many experiments have been performed with the goal of detecting laughter (e.g. [11,12]), and this task might prove useful in emotion recognition and in general man-machine interactions. Apart from laughter, the detection of filler events has also become popular (e.g. [14,18]). Besides serving to regulate the flow of interaction in discussions, it was also shown that filler events are an important sign of hesitation; hence their detection could prove useful during the automatic detection of various kinds of dementia such as Alzheimer's Disease [10] and Mild Cognitive Impairment [18].

In the tasks of detecting laughter and filler events, it is well known (see e.g. [3,6,9]) that although classification and evaluation are performed at the frame

A. Ronzhin et al. (Eds.): SPECOM 2016, LNAI 9811, pp. 232–239, 2016.
DOI: 10.1007/978-3-319-43958-7_27

level, it is worth making use of the contextual information and adjusting the frame-level scores based on the local neighborhood. Gupta et al. [9] applied probabilistic time series smoothing; Brueckner et al. [3] trained a second neural network on the output of the first, frame-level one to smooth the resulting scores; while Gosztolya [6] used the Simple Exponential Smoothing method on the frame-level posterior likelihood estimates.

What is common in these studies is that first they trained a frame-level classifier such as Deep Neural Networks (DNN) to detect the given phenomena, and then, as a second step, they aggregated the neighbouring posterior estimates to get the final scores. It is not clear, however, what type of smoothing may prove to be optimal. In this study we compute the weighted mean of the neighbouring scores as a time series smoothing filter; but even with this type of aggregation, the optimal weight values have to be determined. We treated this task as an optimization one in the space of frame-level weights, where we seek to maximize the Area Under the Curve (AUC) score for the phenomena we are looking for (now laughter and filler events). To find the optimal weight values, we applied genetic algorithms. Using the optimal filters found on the development set, we significantly outperformed both the unsmoothed ("raw") values and some standard time series filters of the same size on the test set of a public English dataset containing laughter and filler events.

2 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive methods which may be used to solve search and optimization problems [1]. The concept and mechanisms applied are based on the genetic processes of biological organisms. Mimicking the evolution of biological populations by selection and recombination, genetic algorithms are able to "evolve" solutions to real world problems.

Genetic Algorithms use a direct analogy of natural behaviour. They work with a *population* of *individuals*, each representing a valid solution to the given problem. Each individual consists of a number of parameters (*genes*). To each individual, a *fitness score* is assigned, which is based on how good a solution it is to the given task. Individuals with higher fitness scores are given opportunities to "reproduce" by "cross breeding" (*crossover*) with other individuals in the population. In this way new individuals are generated that share some features taken from each parent. Then, to each child, *mutation* is applied, which usually means that with a small probability (e.g. 0.001), a gene is changed to some random value. The traditional view is that, from the two recombination steps, crossover is the more important one for rapidly exploring the search space, but mutation provides a small amount of random search [1]. The individuals constructed in this way will form the population of the next *generation*. Traditional GAs start from a randomly generated population and repeat the above steps for several generations. Lastly, the best solution will be the individual in the last population with the highest fitness score.

3 Experimental Setup

3.1 The SSPNet Vocalization Corpus

The SSPNet Vocalization Corpus [14] consists of 2763 short audio clips extracted from telephone conversations of 120 speakers, containing 2988 laughter and 1158 filler events. We used the feature set provided for the Interspeech 2013 Computational Paralinguistics Challenge (ComParE, [16]). It consisted of the frame-wise 39-long $MFCC + \Delta + \Delta\Delta$ feature vector along with voicing probability, HNR, F0 and zero-crossing rate, and their derivatives. To these 47 features their mean and standard derivative in a 9-frame long neighbourhood were added, resulting in a total of 141 features [16]. Each frame was labeled as one of three classes, namely "laughter", "filler" or "garbage" (meaning both silence and non-filler non-laughter speech).

We followed the standard routine of dividing the dataset into training, development and test sets published in [16]. As evaluation metrics, we used the method of evaluation which is the de facto standard for laughter detection: we calculated the Area Under Curve (AUC) score for the output likelihood scores of the class of interest. As we now seek to detect two kinds of phenomena (laughter and filler events), we calculated AUC for both social signals; then these AUC values were averaged, giving the Unweighted Average Area Under Curve (UAAUC) score [16].

3.2 Frame-Level Classification

Before applying a time series filter, first we have to somehow get a likelihood estimate for each class and frame of the utterances. For this, we utilized two state-of-the-art machine learning methods, which we will briefly describe below.

AdaBoost.MH. AdaBoost.MH [15] is an efficient meta-learner algorithm, which seeks to build a strong *final classifier* from a linear combination of simple, scalar *base classifiers*. For more complex problems, the state-of-the-art performance of AdaBoost.MH is usually achieved using *decision trees* as base learners, parametrized by their number of leaves.

We utilized an open source implementation (the *multiboost* tool [2]), and followed a multi-armed bandit (MAB) setup, which can speed up training significantly. In it, for each boosting iteration step, the optimal base learner is found using only a small subset of features, and the usefulness of these subsets are learned from the accuracy of these basic classifiers [4]. We sampled the over-represented "garbage" class, and included the feature vectors of 8 neighbouring frames on each side. We then used 8-leaved decision trees as base learners, and trained our model for $100,000$ iterations. For the details, see [7].

Deep Rectifier Neural Networks. Deep neural networks differ from conventional ones in that they consist of several hidden layers. This deep structure can

provide significant improvements in results compared to earlier techniques used, but the conventional backpropagation algorithm has problems when training such networks. For this, one possible solution is deep rectifier neural networks [5].

In deep rectifier neural networks, rectified linear units are employed as hidden neurons, which apply the rectifier activation function $\max(0, x)$ instead of the usual sigmoid one [5]. The main advantage of deep rectifier nets is that they can be trained with the standard backpropagation algorithm, without any tedious pre-training (e.g. [8]). We used our custom implementation, originally developed for phoneme classification. On the TIMIT database, frequently used as a reference dataset for phoneme recognition, we achieved the best accuracy known to us [17].

For the actual task, we trained our model on 31 consecutive neighbouring frame vectors. (Due to shorter execution times, we were able to carry out more experiments with neural networks than with AdaBoost.MH.) After preliminary tests, we used five rectified hidden layers, each consisting of 256 neurons, and we had neurons that used the softmax function in the output layer.

3.3 Frame-Level Likelihood Aggregation

After obtaining the frame-level likelihood estimates of our classifiers (the "raw" scores), in the next part we will aggregate the values in the local neighbourhood in order to improve the AUC scores. We chose the weighted form of the moving average time series filter; that is, for a filter with a width of $2N + 1$ we define the weight values as $w_{-N}, w_{-N+1}, \ldots, w_N \geq 0$ and $\sum_{i=-N}^{N} w_i = 1$. Afterwards, for the jth frame with the raw likelihood estimate a_j we calculate

$$a'_j = \sum_{i=-N}^{N} w_i a_{j+i}. \tag{1}$$

(Here we used the simplification that, for an utterance consisting of k frames, $a_j = a_1$ for $\forall j \leq 0$, and $a_j = a_k$ for $\forall j > k$.) We then optimized the w_i weight values using genetic algorithms. To test whether the (possible) improvements in the AUC scores come from the actual weight vector and not from the fact that we use some kind of aggregation, we also tested two simple approaches. In the first one, we took the unweighted average of the raw likelihood estimates; that is, we had $w_i = \frac{1}{2N+1}$ (*constant* filter). In the second approach we randomly generated the w_i values (*random* filter).

3.4 Applying Genetic Algorithms

We represented each time series filter by a vector of the w_i weights (i.e. each gene corresponded to a w_i weight). We used filters of size of 129 (64 frames at both sides), based on the results of preliminary tests. We supposed that the optimal weights of the neighbouring frames are not completely independent of each other, so we only stored one weight for every eight frames, while we linearly interpolated

Table 1. The AUC scores for the laughter and filler events got by using the different classification and aggregation methods.

ML method	Filter type	Dev. set			Test set		
		Lau.	Fil.	Both	Lau.	Fil.	Both
AdaBoost.MH	—	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
	Genetic alg.	**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
	Genetic alg.	96.7	**96.5**	**96.6**	94.3	**88.8**	**91.6**
DNN + Prob. time series smoothing [9]		95.1	94.7	94.9	93.3	89.7	91.5
DNN + DNN [3]		98.1	96.5	97.3	94.9	89.9	92.4
ComParE 2013 baseline [16]		86.2	89.0	87.6	82.9	83.6	83.3

the weight values for the intermediate frames. This approach resulted in a more compact weight vector (only 17 values overall instead of 129), which should be easier to optimize.

We utilized the JGAP Java Genetic Algorithm Package [13]. The population size was 250, while we evolved for 100 generations. We used averaging crossover, while for mutation we randomly changed the value of one weight in the weight vector (with the default probability value of 0.001). To keep the frame weight values on the same scale, for each step we normalized each weight vector so that the weights summed up to one. We optimized the filter of the laughter and the filler phenomena independently; the fitness function was the AUC score of the given phenomenon on the development set.

4 Results

Table 1 lists the output AUC and UAAUC scores we got for the two classifier methods and the time series filter approaches. The first thing to notice is that the raw scores (indicated by the "—" filter type) are quite competitive, compared to the ComParE baseline, which were not smoothed over time either. As for the two classifier methods, AdaBoost.MH performed somewhat better; the reason for this is probably that we sampled the database during classifier model training, therefore the distribution of the three classes (that of garbage, laughter and filler events) was more balanced, resulting in more precise likelihood estimations for the laughter and filler classes.

Upon examining the two basic smoothing approaches used for reference (filters "random" and "constant"), we can see that applying these approaches alone brings a significant improvement over the raw likelihood scores. This indicates

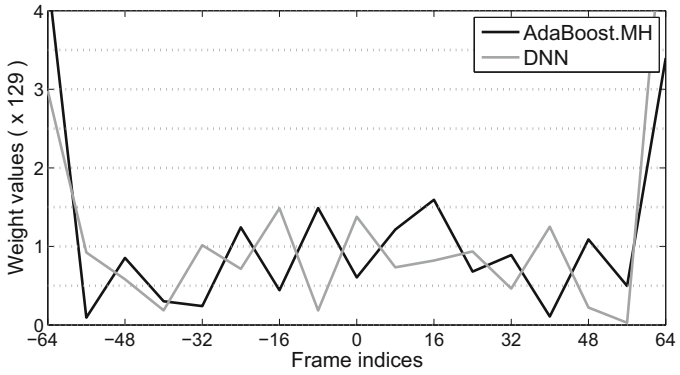

Fig. 1. The optimal filters found using the genetic algorithm for laughter events

that just by utilizing a smoothing filter of this width (which is over a second long) we can noticeably improve the AUC values of the likelihood estimates. Over these scores, however, the weight vectors optimized by the genetic algorithm gives an additional 1 % gain in AUC on the test set, which, in our opinion, justifies our approach of utilizing a weighted average time series filter over the raw likelihood estimates, and optimizing the weights using genetic algorithm. Of course, the width of the filter has to be set carefully, which requires further investigation, just as the number of weights in the weight vector (recall that now for each eight frame we optimized only one weight, while the remaining ones were linearly interpolated). Furthermore, the application of other crossover operators besides the averaging one (for example single point crossover, or a crossover operator which takes the mean of neighbouring filter weight values, therefore smoothing the whole time series filter) could be tested as well, but this falls outside the scope of this study.

4.1 The Time Series Filters Found

Figures 1 and 2 show the time series smoothing filters got by using a genetic algorithm for the laughter and filler events, respectively. The weight values were scaled up to 129 times for better readability (i.e. a weight value of 1 means average importance for the given frame). The large straight sections are due to the linear interpolation of the intermediate frames. It can be seen that the filters are not really smooth themselves, which is probably due to the optimization technique used. Despite this, the two filters belonging to the two different machine learning methods are quite similar to each other for both phenomena.

The filters found for the laughter events have slightly higher weight values around the central frame than those further away (although this tendency is disturbed by the noise present in the weight vectors, which is probably due to the random population initialization of GA). However, what is quite interesting is that the first and last weight values for both machine learning methods are quite

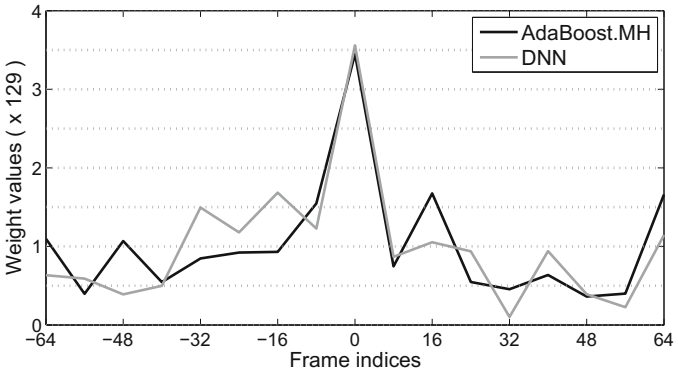

Fig. 2. The optimal filters found using the genetic algorithm for filler events

high, being 3–4 times the average weight. For an explanation of this phenomenon recall that our classifier models were trained using the feature vectors of the 8-8 and 15-15 neighbouring frames on both sides for AdaBoost.MH and DNNs, respectively. This means that the posterior estimate provided by a DNN for the first frame in the smoothing filter already includes some information about the 15 preceding frame, and using the likelihood estimate of the last frame we can "peek" into the 15 consecutive frames. This makes the first and last frames in the averaged filter more important than the inner ones, while the values of the inner frames are redundant to some extent.

This effect is present on the time series smoothing filters found for the filler events, although surprisingly only at the end of the filter. Here, however, the middle frames seem to be very important, having a relative importance of about 3.5 times that of an average frame. This holds for the filters found for both machine learning methods.

5 Conclusions

In this study, we investigated the task of laughter and filler detection. As was shown earlier, after some frame-level posterior estimation step performed via some machine learning method, it is worth smoothing the output likelihood scores of the consecutive frames; so we applied a weighted averaging time series smoothing filter. To set the weights in the filter, we applied genetic algorithms, using the development set of a public English dataset. Our AUC scores got on the test set significantly outperformed both the unsmoothed likelihood values and standard time series filters of the same size.

Acknowledgments. The Titan X graphics card used for this research was donated by the NVIDIA Corporation.

References

1. Beasley, D., Bull, D.R., Martin, R.R.: An overview of genetic algorithms: part 1, fundamentals. Univ. Comput. **15**(2), 58–69 (1993)
2. Benbouzid, D., Busa-Fekete, R., Casagrande, N., Collin, F.D., Kégl, B.: MultiBoost: a multi-purpose boosting package. J. Mach. Learn. Res. **13**, 549–553 (2012)
3. Brueckner, R., Schuller, B.: Hierarchical neural networks and enhanced class posteriors for social signal classification. In: Proceedings of ASRU, pp. 362–367 (2013)
4. Busa-Fekete, R., Kégl, B.: Fast boosting using adversarial bandits. In: Proceedings of ICML, vol. 27, pp. 143–150 (2010)
5. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier networks. In: Proceedings of AISTATS, pp. 315–323 (2011)
6. Gosztolya, G.: On evaluation metrics for social signal detection. In: Proceedings of Interspeech, Dresden, Germany, pp. 2504–2508, September 2015
7. Gosztolya, G., Busa-Fekete, R., Tóth, L.: Detecting autism, emotions and social signals using AdaBoost. In: Proceedings of Interspeech, Lyon, France, pp. 220–224, August 2013
8. Tóth, L., Grósz, T.: A comparison of deep neural network training methods for large vocabulary speech recognition. In: Habernal, I. (ed.) TSD 2013. LNCS, vol. 8082, pp. 36–43. Springer, Heidelberg (2013)
9. Gupta, R., Audhkhasi, K., Lee, S., Narayanan, S.S.: Speech paralinguistic event detection using probabilistic time-series smoothing and masking. In: Proceedings of InterSpeech, pp. 173–177 (2013)
10. Hoffmann, I., Németh, D., Dye, C., Pákáski, M., Irinyi, T., Kálmán, J.: Temporal parameters of spontaneous speech in Alzheimer's disease. Int. J. Speech-Language Pathol. **12**(1), 29–34 (2010)
11. Knox, M.T., Mirghafori, N.: Automatic laughter detection using neural networks. In: Proceedings of Interspeech, pp. 2973–2976 (2007)
12. Neuberger, T., Beke, A., Gósy, M.: Acoustic analysis and automatic detection of laughter in Hungarian spontaneous speech. In: Proceedings of ISSP, pp. 281–284 (2014)
13. Rotstan, N.: JGAP: Java Genetic Algorithms Package (2005). http://jgap.sourceforge.net/
14. Salamin, H., Polychroniou, A., Vinciarelli, A.: Automatic detection of laughter and fillers in spontaneous mobile phone conversations. In: Proceedings of SMC, pp. 4282–4287 (2013)
15. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
16. Schuller, B., Steidl, S., Batliner, A., Vinciarelli, A., Scherer, K., Ringeval, F., Chetouani, M., Weninger, F., Eyben, F., Marchi, E., Salamin, H., Polychroniou, A., Valente, F., Kim, S.: The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism. In: Proceedings of Interspeech (2013)
17. Tóth, L.: Phone recognition with hierarchical convolutional deep maxout networks. EURASIP J. Audio, Speech, Music Process. **2015**(25), 1–13 (2015)
18. Tóth, L., Gosztolya, G., Vincze, V., Hoffmann, I., Szatlóczki, G., Biró, E., Zsura, F., Pákáski, M., Kálmán, J.: Automatic detection of mild cognitive impairment from spontaneous speech using ASR. In: Proceedings of Interspeech, Dresden, Germany, pp. 2694–2698, September 2015