INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Determining Native Language and Deception Using Phonetic Features and Classifier Combination

Gábor Gosztolya[1,2], *Tamás Grósz*[1], *Róbert Busa-Fekete*[3], *László Tóth*[2]

[1]Institute of Informatics, University of Szeged, Hungary
[2]MTA-SZTE Research Group on Artificial Intelligence, Szeged, Hungary
[3]Department of Computer Science, University of Paderborn, Germany
{ ggabor, groszt, tothl } @ inf.u-szeged.hu, busarobi@upb.de

Abstract

For several years, the Interspeech ComParE Challenge has focused on paralinguistic tasks of various kinds. In this paper we focus on the Native Language and the Deception sub-challenges of ComParE 2016, where the goal is to identify the native language of the speaker, and to recognize deceptive speech. As both tasks can be treated as classification ones, we experiment with several state-of-the-art machine learning methods (Support-Vector Machines, AdaBoost.MH and Deep Neural Networks), and also test a simple-yet-robust combination method. Furthermore, we will assume that the native language of the speaker affects the pronunciation of specific phonemes in the language he is currently using. To exploit this, we extract phonetic features for the Native Language task. Moreover, for the Deception Sub-Challenge we compensate for the highly unbalanced class distribution by instance re-sampling. With these techniques we are able to significantly outperform the baseline SVM on the unpublished test set.

Index Terms: accent recognition, SVM, deep neural networks, AdaBoost.MH, classifier combination, instance sampling

1. Introduction

Computational paralinguistics, a subfield of speech technology, is concerned with the non-linguistic information content of the speech signal. A large number of different paralinguistic tasks exist like detecting laughter [1, 2, 3], emotions [4, 5], estimating the intensity of conflicts [6, 7, 8], and so on. The importance of this area is reflected in the fact that for several years now the Interspeech Computational Paralinguistic Challenge (ComParE) has been held regularly (e.g. [9, 10, 11]).

Here, we describe our approach for the Deception and the Native Language sub-challenges of ComParE 2016 [12]. In the first sub-challenge, deceptive speech has to be identified, while in the Native Language Sub-Challenge, the task is to determine the native language (referred to as L1) of the speaker while he is speaking in another language (L2), this time in English. Following the Challenge guidelines (see [12]), we will omit the description of the tasks, datasets and the method of evaluation, and concentrate on the techniques we applied. We should also note that, unlike in a standard conference study, in this case it makes sense to experiment with several techniques at the same time, which we will indeed do.

Determining the L1 language of the speaker (or *accent recognition*) is a well-studied task within speech technology (see e.g. [13, 14, 15]). Similarly to the standard techniques found in the literature, our approach for the Native Language Sub-Challenge is based on the fact that L1 affects the pronunciation of L2 phonemes. However, to avoid the complexity introduced by i-Vectors, which are the standard solution for this task, we first performed frame-level phoneme identification. Then we examined the frame-level DNN outputs, and extracted different features based on them, as we assumed that they encode valuable information on the pronunciation of the L2 phonemes.

After feature extraction, the next important step is that of classification. However, there are several machine learning algorithms available, which can be considered as state-of-the-art. In our study we examined three such methods, namely Support-Vector Machines (SVM, [16]), DNN and AdaBoost.MH [17]. We also applied a robust classifier output combination method, and, for the Deception Sub-Challenge, instance re-sampling.

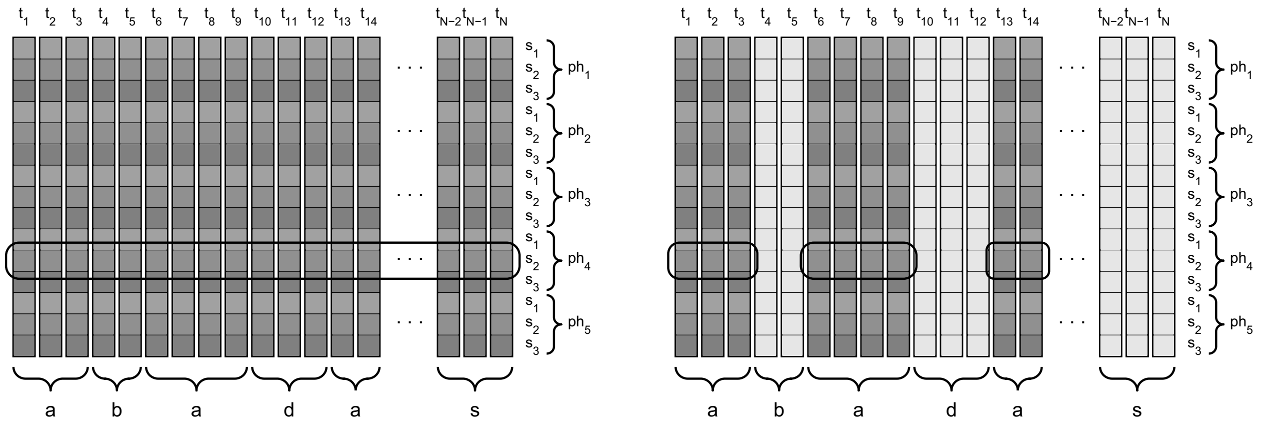
The structure of this paper is as follows. First, in Section 2 we describe the way we extracted the phonetic features of the utterances of the Native Language Sub-Challenge, and we also present the results got with these features on the development set. Next, in Section 3 we describe the classifier methods used, how we set their hyper-parameters, the way we combined their outputs, and we then describe the sampling techniques applied. Lastly, in Section 4.2 we present and analyze our test results.

2. Phonetic Feature Extraction from DNN Output for Native Language Determination

Our approach for identifying the native language of the speaker (i.e. Native Language Sub-Challenge) was based on the observation that native language significantly affects the pronunciation of certain L2 phonemes. In speech recognition terms it means that the L1 languages of the speakers may lead to further errors in the ASR output. However, as the reason for these mispronunciations is partly that the speaker has a different native language, they supposedly appear as some tendencies in the phoneme confusion matrix of the phoneme-level ASR output.

Of course, to construct a phoneme confusion matrix we would need a ground truth transcription of the utterances, which was not available (and in an application situation it is expected to be unknown anyway). However, assuming that, in the long term, phonemes in a given (L2) language follow some specific distribution, we can expect that the phonemes in the ASR output will differ from this ideal distribution (that can be obtained from native speakers), and this tendency is L1-dependent.

To this end, first we performed speech recognition on the utterances, using a DNN/HMM hybrid model. This resulted in a phoneme-level posterior probability vector for each frame, and a time-aligned phoneme sequence for each utterance, which served as a basis for the next feature extraction step.



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{N-2} t_{N-1} t_N
s_1 s_2 s_3
ph_1
ph_2
ph_3
ph_4
ph_5
a
b
a
d
a
s

Figure 1: *Feature extraction from the frame-level output of the CI DNN. The vertical bars represent the CI DNN output vectors for frame t_i, belonging to the states $s_1 \dots s_3$ of each possible phoneme ph_j. The circled region displays the region for averaging: in the "DNN stats #1" strategy (left hand side) we take the mean of the frames of the whole utterance, while in the "DNN stats #2" strategy (right hand side) we average out the frames belonging to the same phoneme, according to the time-aligned ASR output.*

2.1. Obtaining a Time-Aligned Phoneme Sequence by ASR

As the acoustic model we utilized a Deep Rectifier Neural Network [18] with 5 hidden layers and 1000 neurons in each layer. The DNN was trained on the fairly large, freely available "TEDLIUM" English speech corpora [19], following the Kaldi recipe [20]. We applied our custom DNN implementation for GPU, which achieved outstanding results on several datasets (e.g. [21, 22]). As acoustic features 12 mel-frequency spectral coefficients ("MFCC", [23]), along with energy, and their first and second order derivatives were used. We used context-independent (CI) phoneme models, based on the results of our preliminary tests. We had 47 phonemes, which, since we used a tri-state setup, led to 141 states overall.

2.2. Feature Extraction

Next, we extracted numerous feature sets from the time-aligned phonetic ASR output. Note that we implicitly made use of the fact that the utterances are of roughly the same length, hence fairly reliable statistics can be calculated from them. Furthermore, besides our feature sets extracted, we also utilized the 6373-item feature set provided by the Challenge organizers (see [12]), which will be referred to as the standard feature set.

The simplest feature set that we constructed consisted of the total number of occurrences and the total duration of occurrences for each phoneme (feature set of "phoneme stats"). With the 47-item phoneme set, this resulted in a quite compact vector with only 94 attributes for each utterance.

We based the following two feature sets on the frame-level output of our acoustic CI tri-state DNN. In the first feature set (referred to as "DNN stats #1"), we calculated the mean and standard deviation of each DNN output utterance-wise, resulting in 282 features overall. This is also a quite compact feature set compared to the 6373-long standard feature vector. Lastly, we calculated the mean and the standard deviation of the DNN outputs, but this time we also used the time-alignment we got previously: we calculated these scores for the frames belonging to the same phoneme in the phoneme-level ASR output ("DNN stats #2"). This led to 13254 attributes overall. The scheme of these feature extraction steps can be seen in Figure 1 above.

Feature set	Size	Acc.	UAR
Standard	6 373	46.9%	47.1%
Phoneme stats	94	45.2%	45.5%
DNN output stats #1	282	54.7%	54.8%
DNN output stats #2	13 254	56.7%	56.9%
Standard + phoneme stats	6 467	50.0%	50.2%
Standard + DNN stats #1	6 655	53.4%	53.5%
Standard + DNN stats #2	19 627	61.1%	61.3%
All features	20 003	62.9%	63.1%
ComParE baseline [12]		44.9%	45.1%

Table 1: *Accuracy and UAR scores got by using the different feature sets with SVM applied on the development set of the Native Language Sub-Challenge.*

2.3. Results

We evaluated the different feature sets by applying Support-Vector Machines (SVM, [16]) with a linear kernel, using the libSVM library [24]. The value of C was tested in the range $10^{\{-5,\dots,1\}}$. The results for this can be seen in Table 1 above. Our scores for the standard, 6373-item feature set are slightly above the baseline scores provided in the Challenge paper for two reasons: we used libSVM instead of Weka, and we performed an energy-based volume normalization on the input utterances.

What was quite surprising is that by just using the total number and duration of the phonemes (only 94 attributes overall), we managed to slightly outperform the baseline accuracy scores, and almost match our scores got by using the standard, 6373-long feature set. This is really unexpected for such a compact and simple feature set, and in our opinion it indicates that attempting to identify the native language of the speaker based on the phonetic-level ASR output is a viable approach. By using the features extracted from the frame-level DNN outputs, we even managed to significantly outperform the baseline scores; and by combining them with the standard feature set we achieved accuracy scores significantly above the baseline score. In fact, combining all the above-mentioned features led to a UAR score of 63.1% on the development set, which means a 32% relative error reduction compared to the baseline.

3. Classification Methods and Refinements

After describing the feature extraction step, we turn to the next phase: that of classification. Besides describing the classification algorithms utilized for both sub-challenges, we will explain the other techniques applied: DNN model output aggregation, the sampling methods and a classifier combination approach.

3.1. Classifier Methods

We utilized three methods for classification. First, we applied Support-Vector Machines [16] with a linear kernel. The value of C was again tested in the range $10^{\{-5,\ldots,1\}}$. Then for both tasks we trained 10 randomly initialized Deep Rectifier Neural Networks (DNN, [18]) having three fully connected hidden layers, each containing 100 and 500 neurons for the Deception and the Native Language sub-challenges, respectively. We used our custom implementation, originally developed for phoneme classification, by which we achieved a good performance in the previous ComParE Challenges [22, 25]. Finally, we applied AdaBoost.MH [17, 26] using stumps as base learners, because we also got good results previously with it (e.g. [5, 25, 27]). The meta-parameters of each method (C for SVM, number of iteration for AdaBoost.MH) were set using the development set for both sub-challenges.

3.2. DNN Model Output Aggregation

Training a neural network is a non-deterministic procedure due to the random initial weight values. To reduce this effect of uncertainty, it is common to train several models with the same parameters, and aggregate their outputs in some way. Perhaps the most commonly used way of aggregating outputs is via simple majority voting: we choose the class label which was supported by the largest number of models. It can be readily applied to neural networks, but it is well known that DNNs are able to produce accurate posterior scores; unfortunately, this information is lost during simple majority voting. So we used the technique called *probabilistic voting* [25]: for each example and each class we averaged out the output posterior values of all models, and chose the class where this value was the highest.

3.3. Instance Re-Sampling

Most machine learning algorithms are sensitive to class imbalances, and tend to behave inaccurately on classes having only a few examples. Since in the Deception Sub-Challenge the distribution of the D and ND classes was unbalanced to the extent that even the baseline included the upsampling of the D class (i.e. using the same training examples several times), we decided to experiment with sampling methods for this sub-challenge. For DNN, we applied the sampling method called probabilistic sampling [28, 29]. It is a simple two-step sampling scheme: first we select a class, then randomly pick a training sample from the samples of this class. Selecting a class can be viewed as sampling from a multinomial distribution after we assign a probability to each class:

$$P(c_k) = \lambda \frac{1}{K} + (1-\lambda) P_0(c_k), \tag{1}$$

where $P_0(c_k)$ is the prior possibility of class c_k, K is the number of classes and $\lambda \in [0, 1]$ is a parameter. If λ is 1, then we get a uniform distribution over the classes, and with $\lambda = 0$ we get the original class distribution. Choosing a value between 0 and 1 for λ allows us to linearly interpolate between these two distributions.

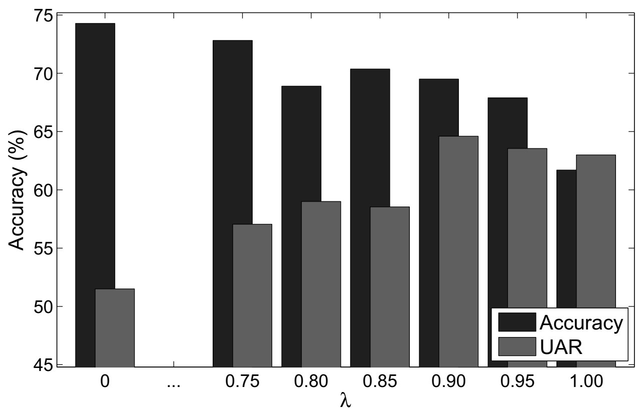


Figure 2: *Accuracy and UAR scores got on the development set of the Deception Sub-Challenge for DNN probabilistic sampling.*

The value of the λ parameter was set on the development set. The accuracy and UAR scores we got can be seen in Fig. 2. It is clear that DNNs optimize for accuracy by default ($\lambda = 0$); by increasing the value of λ, the UAR scores tend to rise, while accuracy tends to drop. In the end we used the optimal value of $\lambda = 0.9$ in our later experiments.

For SVM and AdaBoost.MH, we did not have the option of using probabilistic sampling, as we utilized standard libraries (libSVM [24] and multiboost [26]). Therefore we experimented both with upsampling and downsampling (i.e. not using all the training instances) to balance the class distribution. We trained several models for AdaBoost.MH, so we selected the training set randomly for each training.

M.L. Method	Sampling	Acc.	UAR
SVM	—	69.8%	58.1%
	Upsampling	67.9%	62.6%
	Downsampling	62.8%	62.8%
DNN	—	74.3%	51.5%
	Probabilistic	69.5%	64.7%
AdaBoost.MH	—	72.6%	57.1%
	Upsampling	72.2%	57.3%
	Downsampling	66.7%	62.5%
ComParE baseline [12]		70.2%	61.1%

Table 2: *Accuracy and UAR scores got by using the different classifier and sampling methods on the development set of the Deception Sub-Challenge.*

Table 3 shows the accuracy and UAR scores achieved by using the different re-sampling strategies. It is cleat that re-sampling greatly improved the performance of each method; it was a bit surprising, though, that for AdaBoost.MH, downsampling (i.e. discarding training examples) worked better. Although for SVM, downsampling was slightly better that upsampling UAR-wise, due to the much higher accuracy score we decided that we should use upsampling later. In the following experiments carried out on the Deception Sub-Challenge, we always applied instance re-sampling during classifier model training.

Method	Dev.		Test	
	Acc.	UAR	Acc.	UAR
SVM (upsampling)	67.9%	62.6%	—	—
DNN (prob. sampling)	69.5%	**64.7%**	69.8%	**68.6%**
AdaBoost.MH (downs.)	66.7%	62.5%	—	—
SVM + DNN	69.3%	64.0%	**70.6%**	67.7%
SVM + AdaBoost.MH	67.5%	62.8%	—	—
DNN + AdaBoost.MH	66.9%	61.9%	—	—
All three methods	67.7%	62.9%	—	—
ComParE baseline [12]	**70.2%**	61.1%	—	68.3%

Table 3: *Accuracy and UAR scores got by using the different classifier methods on the Deception Sub-Challenge.*

Method	Dev.		Test	
	Acc.	UAR	Acc.	UAR
SVM	62.9%	63.1%	—	—
DNN	64.5%	64.5%	62.9%	62.9%
AdaBoost.MH	69.3%	69.3%	69.2%	69.2%
SVM + DNN	64.5%	64.7%	—	—
SVM + AdaBoost.MH	70.1%	70.1%	—	—
DNN + AdaBoost.MH	**70.7%**	**70.7%**	**70.0%**	**70.1%**
All three methods	69.0%	69.1%	—	—
ComParE baseline [12]	44.9%	45.1%	—	47.5%

Table 4: *Accuracy and UAR scores got by using the different classifier methods on the Native Language Sub-Challenge.*

3.4. Classifier Combination

It is well known that a good combination of the original classifiers may reinforce their advantages. This may explain the interest in classifier combination techniques in several areas of Artificial Intelligence (e.g. [30, 31]), and also in speech recognition [32, 33]. In our previous experiments (such as our contribution submitted for the Eating Sub-Challenge for ComParE 2015 [11]), we found that a quite robust way of combining different kinds of classifiers is to aggregate their class-wise posterior scores. We will now also apply this strategy here.

We combined our three classifier methods by taking the mean of their posteriors for each example and class, and chose the class for each example which had the highest posterior score. Getting posteriors is quite easy for SVM; for DNNs we used the aggregated likelihood estimates of several trained models (see Section 3.2). For AdaBoost.MH, we trained 1000 models independently, and for each example we counted how many models voted for each class; these values were then used as posterior estimates after normalizing their sum to one. To calibrate the different distribution of likelihoods for the three methods, we first normalized the vote vectors of the different classification methods so as to have the same standard deviation.

4. Results

4.1. Deception Sub-Challenge

Table 3 shows the values achieved by using the different classifier methods on the development and on the test sets of the Deception Sub-Challenge. We can see that, although we were able to outperform baseline SVM on the development set, this was true for the test set only to a limited extent: with DNN alone, we managed to get a higher UAR value than the baseline by a mere 0.3%, but this is quite different from the 3.6% improvement measured on the development set. And although combining DNN with SVM led to a slight gain in accuracy, it led to a slight drop in the UAR value. In our opinion this can be attributed to the small size and the unbalanced nature of the dataset. Of course, the fact that we used a different SVM implementation and a slightly different instance re-sampling method might also affect the scores we obtained.

4.2. Native Language Sub-Challenge

Table 4 shows the results we got by applying the classifier methods on the development and on the test sets of the Native Language Sub-Challenge. As input we used the extended feature set described in Section 2. Although the construction of this 20003-long feature set was achieved by utilizing SVM, we got better scores on it with DNNs. This is probably because this task is fairly large by computational paralinguistics standards: the training set consists of 3300 utterances, and the development and test sets are roughly 1000 recordings long each. With such a high number of samples a DNN can be trained quite reliably. However, AdaBoost.MH performed even better, and by a large amount. In our opinion this is probably due to the diversity of this feature set. Recall that the 20003 attributes consist of the standard paralinguistic features, phoneme occurrence counts, and means and standard deviations of DNN outputs. Our hypothesis is that different normalization techniques are optimal for these different kinds of attributes, while we always used standardization both for SVM and DNN. AdaBoost.MH, however, does not require any kind of normalization, hence it is not affected by a suboptimal normalization procedure.

As regards classifier combination, we can see that if we combine AdaBoost.MH with either SVM or DNN, the accuracy and UAR scores improve. However, by combining all three methods, these values drop, which is probably due to the fact that, in this case, AdaBoost.MH has only a weight of $1/3$.

Overall, on the test set we observed pretty similar tendencies to those seen on the development set. Even combining DNN with AdaBoost.MH yielded a 1% improvement in both accuracy values. In the end we achieved a UAR value of 70.1%, which is way above the 47.5% score of baseline SVM.

5. Conclusions

In this study, submitted for the Computational Paralinguistic Challenge (ComParE) of Interspeech 2016, we focused on to two classification tasks: the Deception and the Native Language sub-challenges. We utilized three different classifier methods (SVM, DNN and AdaBoost.MH) and also tried out a robust classifier output combination approach. Furthermore, we utilized instance re-sampling techniques for the Deception Sub-Challenge, while we extracted various acoustic features for the Native Language Sub-Challenge. The accuracy scores of our methods revealed that instance re-sampling is essential for achieving competitive scores in the Deception Sub-Challenge. For the Native Language Sub-Challenge, however, we found that our feature extraction approach is a viable one for L1 language determination: with quite basic features such as the number of the different phonemes found in the utterances, we were able to practically match the baseline score got using the standard 6373-long feature vector. By extracting further features we outperformed the baseline method by a large amount even on the unpublished test set, although it seems that the way of normalization significantly affects the accuracy values.

6. References

[1] T. Neuberger and A. Beke, "Automatic laughter detection in spontaneous speech using GMM–SVM method," in *Proceedings of TSD*, 2013, pp. 113–120.

[2] G. Gosztolya, "On evaluation metrics for social signal detection," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 2504–2508.

[3] R. Gupta, K. Audhkhasi, S. Lee, and S. S. Narayanan, "Speech paralinguistic event detection using probabilistic time-series smoothing and masking," in *Proceedings of Interspeech*, 2013, pp. 173–177.

[4] S. L. Tóth, D. Sztahó, and K. Vicsi, "Speech emotion perception by human and machine," in *Proceedings of COST Action*, Patras, Greece, 2012, pp. 213–224.

[5] G. Gosztolya, R. Busa-Fekete, and L. Tóth, "Detecting autism, emotions and social signals using AdaBoost," in *Proceedings of Interspeech*, Lyon, France, Aug 2013, pp. 220–224.

[6] H. Kaya, T. Özkaptan, A. A. Salah, and F. Gürgen, "Random discriminative projection based feature selection with application to conflict recognition," *IEEE Signal Processing Letters*, vol. 22, no. 6, pp. 671–675, 2015.

[7] O. Räsänen and J. Pohjalainen, "Random subset feature selection in automatic recognition of developmental disorders, affective states, and level of conflict from speech," in *Proceedings of Interspeech*, Lyon, France, Sep 2013, pp. 210–214.

[8] G. Gosztolya, "Conflict intensity estimation from speech using greedy forward-backward feature selection," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 1339–1343.

[9] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The Interspeech 2013 Computational Paralinguistics Challenge: Social signals, Conflict, Emotion, Autism," in *Proceedings of Interspeech*, 2013.

[10] B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The INTERSPEECH 2014 computational paralinguistics challenge: Cognitive & physical load," in *Proceedings of Interspeech*, 2014, pp. 427–431.

[11] B. Schuller, S. Steidl, A. Batliner, S. Hantke, F. Hönig, J. R. Orozco-Arroyave, E. Nöth, Y. Zhang, and F. Weninger, "The INTERSPEECH 2015 computational paralinguistics challenge: Nativeness, Parkinson's & eating condition," in *Proceedings of Interspeech*, 2015.

[12] B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The Interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Proceedings of Interspeech*, San Francisco, USA, 2016.

[13] A. DeMarco and S. Cox, "Native accent classification via i-vectors and speaker compensation fusion," in *Proceedings of Interspeech*, Lyon, France, Sep 2013, pp. 1472–1476.

[14] V. Hautamäki, S. M. Siniscalchi, H. Behravan, V. M. Salerno, and I. Kukanov, "Boosting universal speech attributes classification with deep neural network for foreign accent characterization," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 408–412.

[15] H. Behravan, V. Hautamäki, and T. Kinnunen, "Factors affecting i-vector based foreign accent recognition: a case study in spoken Finnish," *Speech Communication*, vol. 66, pp. 118–129, 2015.

[16] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[17] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of AISTATS*, 2011, pp. 315–323.

[19] A. Rousseau, P. Deléglise, and Y. Esteve, "TED-LIUM: an Automatic Speech Recognition dedicated corpus," in *Proceedings of LREC*, 2012, pp. 125–129.

[20] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *Proceedings of ASRU*, Hilton Waikoloa Village, Big Island, Hawaii, US, Nov 2011, pp. 1339–1343.

[21] L. Tóth, "Phone recognition with hierarchical Convolutional Deep Maxout Networks," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 25, pp. 1–13, 2015.

[22] T. Grósz, R. Busa-Fekete, G. Gosztolya, and L. Tóth, "Assessing the degree of nativeness and Parkinson's condition using Gaussian Processes and Deep Rectifier Neural Networks," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 1339–1343.

[23] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[24] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[25] G. Gosztolya, T. Grósz, R. Busa-Fekete, and L. Tóth, "Detecting the intensity of cognitive and physical load using AdaBoost and Deep Rectifier Neural Networks," in *Proceedings of Interspeech*, Singapore, Sep 2014, pp. 452–456.

[26] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "MultiBoost: a multi-purpose boosting package," *Journal of Machine Learning Research*, vol. 13, pp. 549–553, 2012.

[27] G. Gosztolya, "Is AdaBoost competitive for phoneme classification?" in *Proceedings of CINTI (IEEE)*, Budapest, Hungary, Nov 2014, pp. 61–66.

[28] L. Tóth and A. Kocsor, "Training HMM/ANN hybrid speech recognizers by probabilistic sampling," in *Proceedings of ICANN*, 2005, pp. 597–603.

[29] S. Lawrence, I. Burns, A. Back, A. Tsoi, and C. Giles, "Chapter 14: Neural network classification and prior class probabilities," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 299–313.

[30] B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, and J.-V. Moreau, "A multi-classifier combination strategy for the recognition of handwritten cursive words," in *Proceedings of ICDAR*, 1993, pp. 642–645.

[31] K. Yu, X. Jiang, and H. Bunke, "Lipreading: A classifier combination approach," *Pattern Recognition Letters*, vol. 18, no. 11–13, pp. 1421–1426, 1997.

[32] L. Felföldi, A. Kocsor, and L. Tóth, "Classifier combination in speech recognition," *Periodica Polytechnica, Electrical Engineering*, vol. 47, no. 1, pp. 125–140, 2003.

[33] G. Gosztolya and J. Dombi, "Applying representative uninorms for phonetic classifier combination," in *Proceedings of MDAI*, Tokyo, Japan, Oct 2014, pp. 182–191.