2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI) | 978-1-6654-9704-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/SAMI54271.2022.9780776

Handling the stochastic behaviour of the Bag-of-Audio-Words method

Mercedes Vetráb
University of Szeged
Szeged, Hungary
vetrabm@inf.u-szeged.hu,
orcid: 0000-0001-7511-2910

Gábor Gosztolya
University of Szeged
Szeged, Hungary
ggabor@inf.u-szeged.hu

***Abstract*—Automatic sleepiness detection may be of help for drivers and air pilots. Sleepiness can be detected from human speech which is a task of paralinguistics. There are several methods available that can overcome this, like the Bag-of-Audio-Words technique. Because it is a stochastic method it can produce different results after each re-run with the same settings. This can be handled with ensembling. Unfortunately the shared disadvantages of BoAW and ensembling, that BoAW needs quite a large feature space to work well and emsembling makes it even larger and larger which, in the end might lead to a lower classification performance. We found that it can be handled by PCA dimension reduction, but there are some edge cases where it does not work well.**

***Index Terms*—Bag-of-Audio-Words, emotion detection, human voice, sound processing, paralinguistics**

I. Introduction

Human speech can encode information about the speaker's physical and mental state (i.e: the emotional state, signs of illness, depression, joy and so on). Computer science and engineering information can use this extra information to create intelligent systems. In certain everyday life situations a communication monitoring system [1], an emotional state detecting system, or a confidence detection system may be useful. Many services use dialogue systems [2] where if they recognise the most problematic dialogue phrases they can work better. In the area of healthcare a mental state monitoring system may be useful for assessing patients [3], [4]. An emotion detection system in call centers also has advantages, such as detecting problematic customers [2]. In our paper we will attempt to detect sleepiness in human speech. There are activities which require concentration and maximum attention from the subject, because if they are fall asleep there will probably be an accident. For example driving, flying and operating machines. Imagine how helpful a system could be that could detect sleepiness and take some preventive action in the case of danger. Human speech can be analysed in many different ways. One of them is by using paralinguistics which involves extracting abstract information from speech recordings.

In paralinguistics the variable length of recordings will always be a significant problem. This is due to the fact that our recordings have different lengths, but our classifiers expect a fixed-sized input. Therefore we have to transform every speech recording into a fixed-sized feature vector. There are several methods that can handle this task. First of all we need a frame-level feature extraction, and afterwards there are many possible ways we can transform these varying length vectors into the same length. For example x-vectors [5], i-vectors [6], Fisher vectors [7], neural networks [8] and the Bag-of-Audio-Words (i.e: BoAW [9], [10]) approach. Here, we investigated the BoAW one. Our experiments were performed on a public German sleepiness database. The BoAW technique can be used effectively in the field of paralinguistics, but it has some drawbacks. One of them is that it uses random numbers during calculations and this affects the results of the extraction and the performance of our classifiers. We tried to handle this with ensembling. It works well, but it increases the feature space and more CPU time is required in the subsequent machine learning step. Hence we tried to compress our ensembled features with PCA. We found that the PCA dimension reduction technique generally works well but it works less well with high dimensions.

II. Bag-of-Audio-Words Method

A possible solution to the problem of varying length is the Bag-of-Audio-Words method. This can be used to create fixed length feature vectors from a different number of frame-level features in recordings. It has a common base algorithm with the Bag-of-Visual-Words [11] and the Bag-of-Words [12] techniques. Now we will introduce the steps of the method.

First, it analyzes the entire database and then it computes fixed length features for each recording. Figure 1 shows the general workflow for generating a BoAW representation for a database. There is an overlap between how it handles our train and test sets. As can be seen, the extraction of the test set depends on the clustering of the train set. Let us now describe the processing of the training set.

The input of the algorithm is the frame-level feature set for each recording that we have. In this case we have a different number of frame-level features for each recording. This number depends on how long the actual recording is and how long the feature window that was used for extraction.

In the first step, it will cluster the feature set. It collects all the frame-level features from all the files and then it puts them into one big "bag". After it performs a clustering on this "bag".

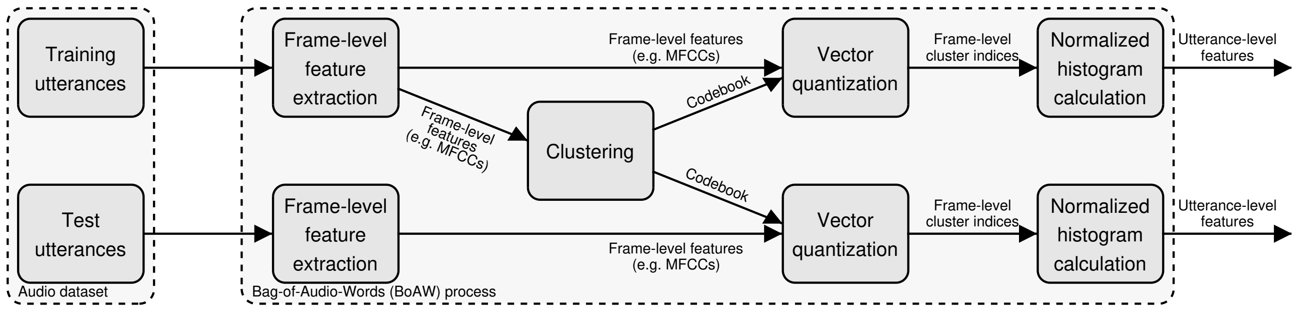
Training utterances
Test utterances
Audio dataset
Frame-level feature extraction
Frame-level feature extraction
Frame-level features (e.g. MFCCs)
Clustering
Codebook
Codebook
Frame-level features (e.g. MFCCs)
Frame-level features (e.g. MFCCs)
Vector quantization
Vector quantization
Frame-level cluster indices
Frame-level cluster indices
Normalized histogram calculation
Normalized histogram calculation
Utterance-level features
Utterance-level features
Bag-of-Audio-Words (BoAW) process

Fig. 1. Workflow for the Bag-Of-Audio-Words technique.

The center vector of the clusters are called "codewords". The group of these "codewords" is the "codebook". The aim of clustering is to separate the original features into meaningful subsets by collects similar vectors into the same group and puts different vectors into different groups. The algorithm can be parametrized, so we can set the number of the clusters (i.e: N). The N parameter is called the codebook size. The vector dimension of the final classification will depend on this codebook size.

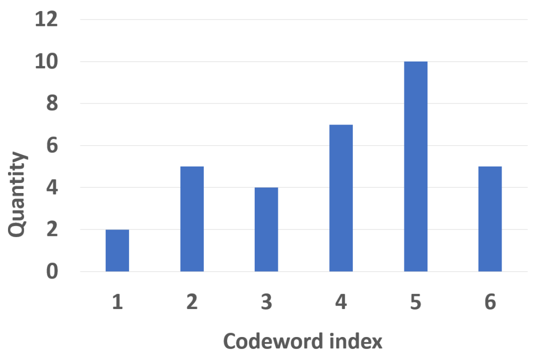

Fig. 2. A Bag-Of-Audio-Words histogram of the recording.

In the next step, the algorithm performs a vector quantization using the previously created clusters. It computes a histogram for each recording independently. It finds the closest "codeword" for each frame-level feature of the recording and replaces the feature with the index of the closest "codeword's" cluster. (As the BoAW algorithm can be parametrized, we can set how many closest "codewords" we wish to find.) Let us call this parameter a. Then it counts how many from the proper index are represented in the given recording. It creates a fixed-sized histogram from these quantities. The x-axis of the histogram represents the index of each "cluster", and the y-axis shows how many of the frame-level feature vectors get mapped into the particular "codeword" of the cluster. For example, Figure 2 shows a histogram of one recording. In this case the codewords are represented by their indices (i.e. $1, 2, 3$ and so on). This recording has 43 frames, and each frame gets mapped into 1 codeword.

In the last step, the algorithm normalizes the computed histograms, as the index quantities are divided by the number of frames of the recording. Lastly, each histogram can be represented by a vector, so the result of this step is a fixed-size feature vector for each recording. The length of these vectors depends on the parameter N, and the vector can be used as a feature set for any classification or machine learning task. The set of these histograms is called the "Bag-of-Audio-Word" representation.

Figure 1 shows how the results of the clustering step can be used in the calculation of the test set representation. The quantization step can be performed with a previously computed "codebook". For example, the frame-level features of the test set came from the same dimension as the train set. So we can define the "codebook" on the train set and then we can quantize the test set features by calculating distances from the previously created "codebook".

A. *Parameters of the BoAW Method*

The BoAW method has many adjustable parameters that can influence the process of codebook creation. In this study, we set the random seed, the preprocessing method, the clustering method, the codebook size N, and the quantization neighbour number parameter a. For the codebook building we used an open-source program called openXBOW [13]. This software allows us to save two important things when we extract the BoAW representation from the train set: the parameter settings applied and the whole codebook. Then we can use these later for the test set.

Random seed: We can set the start-up random seed that is used for codebook creation. This is important because the BoAW method uses random numbers while clustering and these numbers influence the final BoAW representation. In our baseline we automatically generate 10 random seeds and used these random seed values: $1\,964$, 423, $1\,355$, 86, $1\,052$, $1\,549$, 139, 731, 951.

Preprocessing techniques: With openXBOW we can do some preprocessing on the frame-level descriptors, before the clustering step. In our previous study [14], we found that standardization and normalization also improved our results. In order to eliminate the bad effect of outliers we used standardization.

Clustering method: One important factor is the clustering process used to create the codebook. In our previous study [14] we found that the k-means [15] and k-means++ [16] methods both perform well, so in this study we decided to use the k-means++ method.

TABLE I
BASELINE RESULTS WITH THEPEARSON CORRELATION

Codebook size	**Current best**		**Current worst**		**Avg correlation**		**Avg prediction**	
	dev	***test***	***dev***	***test***	***dev***	***test***	***dev***	***test***
32×2	.299	.302	.266	.316	.289	.302	.297	.325
64×2	.310	.329	.277	.311	.292	.324	.309	.334
128×2	.317	.342	.290	.324	.302	.336	.311	.353
256×2	.329	.346	.298	.341	.309	.348	.315	.363
512×2	.325	.370	.311	.366	.316	.365	.320	.368
$1\,024 \times 2$	.328	.372	.316	.372	.323	.374	.326	.377
$2\,048 \times 2$	.332	.383	.325	.379	.328	.379	.331	.382
$4\,096 \times 2$	.334	.382	.321	.378	.327	.379	.330	.384

Histogram neighbour number: Instead of looking for just the closest "codeword", each vector may also be assigned to a certain number of closest "codewords". In our previous study [14], we found that using more than one cluster improved our results, so now we will quantize the 5 closest "codewords". This leads to a more precise description of the recordings with the same feature vector size.

Codebook size: We can control how many clusters we wish to create, which means the length of the feature vectors we would like to create. In each experiment we tested the effect of the following lengths: 32, 64, 128, 256, 512, 1 024, 2 048, 4 096, 8 192.

Derivatives: In speech processing, it is common practice to calculate the first and second derivatives of the frame-level feature vectors. These are the so-called deltas and delta-deltas. These can describe the dynamics of speech [17]. With the help of the openXBOW program, we can create separate codebooks for the original low-level descriptors and another for the Δs. Because of the Δs and the doubled codebooks, every codebook size has to be counted twice, which is included in our figures.

III. DATA AND METHODS

Next, we will present our experimental setup and environment: the database, the classification method and its parameters, the evaluation metric, and the feature set we used.

A. Sleepiness Database

The Sleepy Language Corpus (i.e: Sleepiness database) [18] was introduced in the Interspeech 2011 Speaker State Challenge [19]. It contains 16463 recordings. The recordings came from native German speakers (aged between 20-52 years). One part of it came from a story reading task, another part came from giving verbal commands to the GPS navigator, another from traffic controller communication statements, another from picture descriptions and another from giving a presentation. The dataset was divided into three speaker-disjunct sets as training, development and test sets. The training set contained 20 female and 16 male speakers, 5 564 recordings in total. The development (i.e dev) set contained 17 female and 13 male speakers, and 5 328 recordings in total. The test set contained 19 female and 14 male speakers, 5 571 recordings in total. The labels were defined by a subjective questionnaire that was filled in by the subject and three other assistants. The labels are integers and lie in the range from 1 to 10: extremely alert (1), very alert (2), alert (3), rather alert (4), neither alert nor sleepy (5), some signs of sleepiness (6), sleepy, without any effort to stay awake (7), sleepy, some effort to stay awake (8), very sleepy, great effort to stay awake, struggling against sleep (9), extremely sleepy, cannot stay awake (10).

Previous results for this database produced scores between .260-.383 [20], [21].

B. Feature Set

The feature set used here came from the INTERSPEECH 2013 Paralinguistic Challenge [22]. It contains 65 frame-level features: 55 spectral; 6 voicing related low-level descriptors; 4 energy-related. A 60 ms frame (Gaussian window function) and a sigma value of .4 was used for the speech-related features; and a 25 ms frame (Hamming window function with a step size of 10 ms) for the others.

For feature extraction we used the open-source openSMILE software [23] with the `IS13 ComParE` config file. The final feature set we applied contained not just the basic 65 features, but also their derivatives. We used deltas because we wanted to get information about the dynamics of the speech samples over time.

C. Evaluation Method

The sleepiness database evaluation was carried out as a classification task in the original challenge [20]. Here, we have to predict the factor of the sleepiness on a scale, but we can use only integer numbers. Most of the time when we talk about scales it appears to be more like a regression task than a classification task. Therefore we trained a regression model for prediction calculation like in the original challenge. For the evaluation we used the Pearson correlation [24] and the Spearman correlation like previous submitters did it in the challenge.

The regression was performed using the LIBSVM library [25], which is an SVM (Support Vector Machine [26]) and an SVR (Support Vector Regression Machine [27]) implementation written in many different programming languages. It also has classification and regression models, so we decided to use the epsilon-SVR one. It came as a Python extension. In the training scenario we evaluated it with multiple C complexities in the range 10^{-5} to 10^{-3} (powers of 10: -5; -4; -3).

In our evaluations, the predictions were floating point numbers. Furthermore the mean and the standard deviation of our

TABLE II
BEST RESULTS WITH THE PEARSON/SPEARMAN CORRELATION

	Pearson			**Spearman**		
	dev	**test**	***codebook size***	**dev**	**test**	***codebook size***
Best model	.334	.382	$4\,096 \times 2$	.341	.369	$4\,096 \times 2$
Worst model	.325	.379	$2\,048 \times 2$	.326	.373	$4\,096 \times 2$
Random model	.329	.375	$2\,048 \times 2$	.331	.363	$2\,048 \times 2$
Average model	.328	.378	$2\,048 \times 2$	.331	.369	$4\,096 \times 2$
Prediction average	**.331**	**.382**	$2\,048 \times 2$	**.336**	**.373**	$4\,096 \times 2$
PCA 95	.322	.354	3 319	.322	.328	4 036
PCA 99	.322	.353	4 780	.320	.341	4 780
Challenge original BoAW	–	–	–	.269	.260	2 000
Challenge original majority	–	–	–	–	.341	–
Challenge submitted worst one	–	–	–	.300	.331	–
Challenge submitted best one	–	–	–	.367	.383	–

predictions were very different from the mean and the standard deviation of the original labels. Because of this, we corrected the distribution of our predictions by subtracting their mean from themselves and divided them by their standard deviation. Next, we multiplied them with the standard deviation of the original label set, and added the mean of the original label set. The mean and standard deviation of the original label set came from the dev set. Afterwards, our predictions were still floating point numbers so then we had to round them. We used the standard mathematical rounding, except when the prediction was lower than 1 (then set to 1) or greater than 9 (then we rounded down to 9). Lastly, we calculated the Pearson correlation from these rounded and corrected predictions.

In the case of model training, before each training scenario we applied a Python implemented standardization on the input BoAW feature set.

In the test scenario, we trained a model on the whole training and dev set using the optimal C parameter value found above and evaluated it on the test set.

IV. Tests and Results

In this study, our aim was to reduce the adverse effects of the BoAW algorithm stochasticity. Because BoAW handles random numbers we can have different results with different random seeds and these results have a large deviation.

A. Baseline

In order to test the robustness of the BoAW algorithm, we extracted 10 different BooAW representation and built 10 different models from them. This told us the deviation of the output. The 10 different BoAW representations were extracted with 10 different start-up random seeds. Our baseline came from the results of the following evaluations:

- The run which had the best results on the dev set;
- The run which had the worst results on the dev set;
- We calculated the Pearson correlations for each run and took their average;
- We calculated the sum of 10 predictions got from each run and took their average. Lastly, we calculated the Pearson correlation from them.

The differences between the best predictions may be up to .033 on the dev set and .03 on the test set depending on the codebook size. Table I shows all the results of the baseline cases. (Recalls that codebook size means how many dimensions we have, hence how many features we have. In this case, these are doubled because BoAW creates one codebook from the original features and another one from the delta values.) The best result from the 10 runs was .334 on the dev set and .383 on the test set with a codebook size of 4 096. The worst result from the 10 runs was .325 on the dev set and .379 on the test set with a codebook size of 2 048. The difference between our best and worst baseline results tells us that the BoAW algorithm should be more robust, because if we made only one evaluation, we may just get the worst result. When we collected all the final Pearson correlations from the 10 runs and then calculated their average, our result was .328 on the dev set and .379 on test set with a codebook size of 2 048. When we collected all the final predictions from the 10 runs and got their averages sample by sample and then calculated the Pearson correlation, our result was .331 on the dev set and .382 on the test set with a codebook size of 2 048. Although we calculated the Pearson correlation for each run in two different ways. Table I shows that under the codebook size of 1 024 the result have a difference of about .01, but above this size the difference approximately .003. Table II shows a summary for all the best results of each use-case. For comparison, we also show the results of the original challenge. It can be seen that the original baseline gave .269 on the dev set and .260 on the test set. From the submissions to the original challenge, the worst submission gave .300 on the dev set and .331 on the test set, and the best submission results of the best submission gave .367 on the dev set and .383 on the test set. It can be seen that our method gives better results than the original baseline and it is close to the best submission with .336 on the dev set and .373 on the test set.

B. The Ensemble

To make the BoAW algorithm more robust, we concatenated the BoAW feature vectors from the previous 10 runs. With this concatenation, with only one run we can get the same result as the best result from the baseline. The drawback of this is that our feature space is now 10 times larger. In the case of a codebook size of 32×2 it will be 640, 64×2 it will be 1 280 and so on. Unfortunately it has a major

TABLE III
PCA RESULTS

			Pearson				Spearman			
Concatenated	**Reducted size**		**PCA 95%**		**PCA 99%**		**PCA 95%**		**PCA 99%**	
codebook size	**95**	**99**	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**
640	165	379	.291	.331	.293	.335	.291	.323	.293	.328
1 280	331	753	.302	.337	.302	.338	.297	.332	.297	.333
2 560	700	1 482	.311	.355	.311	.356	.309	.346	.309	.346
5 120	1 369	2 661	.314	.362	.314	.362	.316	.351	.317	.351
10 240	2 337	3 968	.317	.364	.317	.365	.316	.350	.317	.350
20 480	3 319	4 780	.322	.354	.322	.353	.321	.344	.320	.341
40 960	4 036	5 132	.318	.339	.317	.336	.322	.328	.320	.323
81 920	4 473	5 283	.302	.328	.303	.322	.303	.313	.303	.309

drawback on the computing time and the memory used. As we wished to reduce these effects, we ran a Principal Component Analysis (i.e: PCA) on the concatenated feature data. PCA is a dimension-reduction method and it projects the original feature set into a lower dimension space by reducing the number of features. Firstly, it standardizes the dataset. In the second step, it calculates a covariance matrix to see if there is any relationship between the features. In the third step, it finds the principal components in order of significance. Now we know how important our features are by retaining more information after dimension reduction. In the last step we decide on how much information we wish to keep and project our original data using the chosen principal components into the new smaller space. In our investigation we decided to keep 95% and 99% of the original information.

Table III shows the results achieved with dimension reduction. The second and the third columns contain the number of dimensions after the PCA transformation. Keeping $95\% - 99\%$ of the information reduces the original dimension of size to a half/quarter/tenth. The best results with 95% information was .322 on dev set and .354 on the test set. The best results with 99% information was .322 on the dev set and .353 on the test set. As we can see, there was a slight loss in efficiency with compression. An interesting pattern can be seen in the test results. The curve of the test set does not follow the curve of the development set. The concatenated features that were reduced to a third or quarter performed better. We think that this is due to a drawback of the PCA method. It finds linear combinations of the features, but sometimes it fails when the number of features is comparable or even larger than the database size [28]. As we can see in Table III when the concatenated codebook size was more than twice of our recording count ($\approx 5500 \times 2$) the test results started to decrease. Now let us compare the PCA results from lower dimensions where the concatenated codebook size was smaller or equal to the recording size. In this case both of the 95% and the 99% PCA test results are better than the baseline best results. For example, as we can see in Figure 3: codebook size 64 - concatenated size 640 - baseline best test result .302 - PCA 95% test result .332 - PCA 99% test result .335 and so on. This confirms that PCA performs badly when we have more features than samples, but otherwise it performs well.

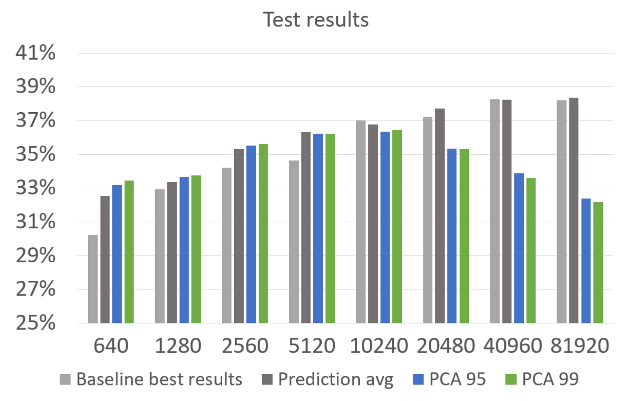

Fig. 3. Test results from the best baseline run and from the PCA evaluations.

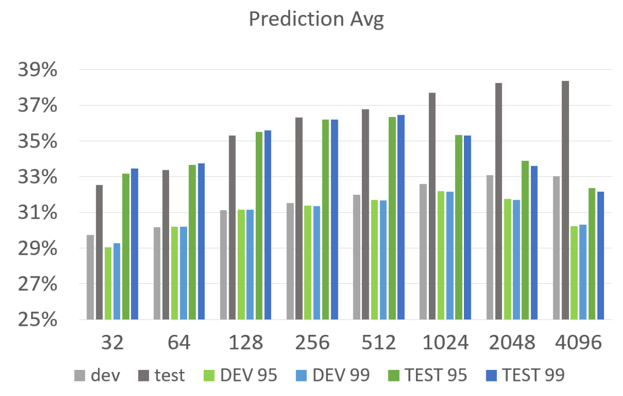

Fig. 4. Test results from the best baseline run and from the PCA evaluations.

V. Conclusion

The goal of our study was to handle the stochastic behaviour of the Bag-of-Audio-Words method, while detecting sleepiness in human speech. We used a German database and we applied the BoAW method for feature extraction. BoAW is a stochastic method because it uses random numbers during the extraction, so it can give us slightly different features for each re-run, even when we use the same settings. We overcame this problem with ensembling. It means we ran the extraction 10 times and concatenated their results. It made the final feature set more robust but it had a big drawback. Unfortunately the BoAW method needs a larger feature space to perform well and emsembling makes it even larger and larger which, in the end might lead to a lower classification performance and excessive CPU time. We found this could be handled by PCA dimension reduction, but there are some edge cases where it does not work well. When we apply PCA on a database that has more features than data, it cannot handle it and gives poor results.

Overall, we may conclude that using the BoAW technique with a paralinguistics task is a good idea, and if we wish to make BoAW more robust we can use the ensemble technique. If the ensemble results in too many dimensions but we still have more data than features, we can reduce the space using PCA.

Acknowledgment

This study was supported by the National Research, Development and Innovation Office of Hungary via contract NKFIH FK-124413. This research was also supported by grant NKFIH-1279-2/2020 of the Ministry for Innovation and Technology, Hungary, and by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program (MILAB). G. Gosztolya was also funded by the János Bolyai Scholarship of the Hungarian Academy of Sciences and by the Hungarian Ministry of Innovation and Technology New National Excellence Program ÚNKP-21-5-SZTE.

References

[1] J. James, L. Tian, and C. Inez Watson, "An open source emotional speech corpus for human robot interaction applications," in *Proceedings of Interspeech*, Hyderabad, India, Sep 2018, pp. 2768–2772.

[2] F. Burkhardt, M. van Ballegooy, K.-P. Engelbrecht, T. Polzehl, and J. Stegmann, "Emotion detection in dialog systems: Applications, strategies and challenges," in *Proceedings of ACII*, Amsterdam, Netherlands, Sep 2009, pp. 985–989.

[3] M. S. Hossain and G. Muhammad, "Cloud-assisted speech and face recognition framework for health monitoring," *Mobile Networks and Applications*, vol. 20, no. 3, pp. 391–399, 2015.

[4] L. Vidrascu and L. Devillers, "Detection of real-life emotions in call centers," in *Proceedings of Interspeech*, Lisbon, Portugal, Sep 2005, pp. 1841–1844.

[5] P. Raghavendra, W. Tianzi, V. Jesus, C. Nanxin, and D. Najim, "X-vectors meet emotions: A study on dependencies between emotion and speaker recognition," 2020.

[6] T. Zhang and J. Wu, "Speech emotion recognition with i-vector feature and RNN model," 07 2015, pp. 524–528.

[7] G. Gosztolya, "Using the Fisher vector representation for audio-based emotion recognition," *Acta Polytechnica Hungarica*, vol. 17, no. 6, pp. 7–23, 2020.

[8] T. Grósz and L. Tóth, "A comparison of Deep Neural Network training methods for Large Vocabulary Speech Recognition," in *Proceedings of TSD*, Pilsen, Czech Republic, 2013, pp. 36–43.

[9] S. Pancoast and M. Akbacak, "Bag-of-Audio-Words approach for multimedia event classification," in *Proceedings of Interspeech*, Portland, USA, Sep 2012, pp. 2105–2108.

[10] M. Schmitt, F. Ringeval, and B. Schuller, "At the border of acoustics and linguistics: Bag-of-Audio-Words for the recognition of emotions in speech," in *Proceedings of Interspeech 2016*, San Francisco, USA, 2016, pp. 495–499.

[11] G. Csurka and F. Perronnin, "Fisher vectors: Beyond bag-of-visual-words image representations," in *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, vol. 229, 01 2011, pp. 28–42.

[12] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, 12 2010.

[13] M. Schmitt and B. Schuller, "openXBOW – Introducing the Passau open-source crossmodal Bag-of-Words toolkit," *Journal of Machine Learning Research*, vol. 18, no. 96, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/17-113.html

[14] M. Vetráb and G. Gosztolya, "Speech emotion detection form a hungarian database with the Bag-of-Audi-Words technique (in hungarian)," in *Proceedings of MSZNY*, Szeged, 2019, pp. 265–274.

[15] S. Pancoast and M. Akbacak, "Softening quantization in bag-of-audio-words," in *Proceedings of ICASSP*, Florence, Italy, May 2014, pp. 1370–1374.

[16] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of SODA*, New Orleans, Louisiana, USA, Jan 2007, pp. 1027–1035.

[17] K. Gergely, "Interactive systems (in Hungarian)," in *Hallgatói Információs Központ*, Budapest, Hungary, 2011.

[18] F. Hoenig, A. Batliner, E. Noeth, S. Schnieder, and J. Krajewski, "Acoustic-prosodic characteristics of sleepy speech - between performance and interpretation," *Proceedings of the International Conference on Speech Prosody*, pp. 864–868, 01 2014.

[19] B. Schuller, A. Batliner, S. Steidl, F. Schiel, and J. Krajewski, "The interspeech 2011 speaker state challenge," 01 2011, pp. 3201–3204.

[20] B. W. Schuller, A. Batliner, C. Bergler, F. B. Pokorny, J. Krajewski, M. Cychosz, R. Vollmann, S.-D. Roelen, S. Schnieder, E. Bergelson, A. Cristia, A. Seidl, A. S. Warlaumont, L. Yankowitz, E. Nöth, S. Amiriparian, S. Hantke, and M. Schmitt, "The INTERSPEECH 2019 Computational Paralinguistics Challenge: Styrian Dialects, Continuous Sleepiness, Baby Sounds & Orca Activity," in *Proc. Interspeech 2019*, 2019, pp. 2378–2382.

[21] G. Gosztolya, "Using fisher vector and Bag-of-Audio-Words representations to identify styrian dialects, sleepiness, baby & orca sounds," in *Proc. Interspeech 2019*, 2019, pp. 2413–2417.

[22] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism," in *Proceedings of Interspeech*, 08 2013, pp. 148–152.

[23] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The Munich versatile and fast open-source audio feature extractor," in *Proceedings of ACM Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 1459–1462. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874246

[24] W. Kirch, "Pearson's correlation coefficient," in *Encyclopedia of Public Health*. Dordrecht: Springer Netherlands, 2008, pp. 1090–1091.

[25] 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.

[26] S. Bernhard, C. P. John, S.-T. John, J. S. Alexander, and C. W. Robert, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[27] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, ser. NIPS'96. Cambridge, MA, USA: MIT Press, 1996, p. 155–161.

[28] I. Johnstone and A. Lu, "On consistency and sparsity for principal components analysis in high dimensions," *Journal of the American Statistical Association*, vol. 104, pp. 682–693, 06 2009.