Mobile App for Stress Monitoring using Voice Features

Virginia Sandulescu[1], Sally Andrews[2], David Ellis[3], Radu Dobrescu[1], Oscar Martinez-Mozos[4]
Affiliation 1: Dept. of Automatic Control and Industrial Informatics, Politehnica University of Bucharest, Bucharest, Romania
Affiliation 2: School of Social Sciences, Nottingham Trent University, Nottingham, UK
Affiliation 3: Department of Psychology, Lancaster University, Lancaster, UK
Affiliation 4: School of Computer Science, University of Lincoln, Lincoln, UK

***Abstract*-The paper describes the steps involved in designing and implementing a mobile app for real time monitoring of mental stress using voice features and machine learning techniques. The app is easy to use and completely non-invasive. It is called StressID and it is available in the Google Play store. With the use of a server application presenting a web interface, interested parties may remotely monitor the stress states detected by the mobile app, enlarging the number of use case scenarios.**

Keywords: stress, SVM, voice features, mobile app.

I. INTRODUCTION

Stress represents the set of reactions of the human body to outside perturbing factors. Stress induces physiological responses like variations in the cardiac rhythm, skin temperature and increases in skin electrodermal activity and also behavioral changes like changes in body postures, mimics or in the way a person speaks.

It has been proven that long term stress correlates with mental health problems like anxiety and depression and physical health problems like heart, cardiovascular diseases and a general poorer immune system [1]. Stress has also been linked with a shorter life hope [2], [3]. Stress may cause health problems by its direct physiological effects or, indirectly, by causing unhealthy habits: inadequate sleeping or eating patterns, smoking, alcohol or drugs abuse [1]. Stress management should begin before it starts causing medical problems. This is where stress monitoring can come in help. The developed mobile app monitors stress using only features extracted from voice and communicates the results to a server application for telemonitoring purposes. This way, a journal of the mental states of the user is available both to the user and also to other potentially interested parties.

II. RELATED WORK

Most of the research in the area is concentrated on identifying stress using physiological parameters. This usually involves wearing specialized devices to acquire the needed data. There are many examples of such works like the ones presented in [4], [5], [6] or [7] and many others. Although using multiple physiological parameters may help in detecting stress with higher accuracies (reported accuracies are, in some cases, higher than 90%), using only voice parameters allows designing systems more easy to use, only requiring the use of microphones that may be integrated in wearable devices. The following paragraphs present the state of the research in identifying stress using voice features.

The work presented in [8] describes a voice processing library for mobile phones, Affective and Mental health MONitor (AMMON). The library is written in C and contains functions for audio signal processing and voice feature extraction: zero crossing rates, pitch, MFCC and several other features. The Support Vector Machine (SVM) algorithm is used for mental states classification. The library is based on a frontend library presented in [9], made by ETSI (European Tele-communication Standards Institute), developed for speech identification. The paper also presents a comparison between AMMON and another library, openSMILE [10] as both of them are used for differentiating between positive and negative emotions using the same database for testing. The performances are similar in terms of accuracy, reaching an average accuracy of 75.51%. openSMILE (The Munich Versatile and Fast Open-Source Audio Feature Extractor is another library created for voice processing. It is developed in C++ and contains functions for audio signal acquisition, processing and feature extraction: signal energy, volume, MFCC, Linear Predicting Coding Coefficients (LPC), pitch, voice formants and other voice features. In [8] a comparison in terms of computing performances of the two libraries is also presented, showing that in certain situations, AMMON may be up to 3 times faster. openSMILE is licensed as open-source software and is free to use for research and personal purposes, while AMMON is not publicly available.

The authors of [11] present a study on stress detection using more than 380 voice features, extracted using openSMILE. It is concluded that the most representative voice features for stress identification are: Mel Frequency Cepstral Coefficients (MFCC), the delta coefficients based on the MFCC, the Teager Energy Operator (TEO) coefficients, voice pitch and zero crossing rates. The highest accuracy achieved is 86%, but the reported average accuracy is 51.13%. The authors report that better accuracies are achieved when different stress models are created for each gender, but the authors of [8] claim that they do not achieve significant differences when using gender-dependent models.

The paper presented in [12] describes a mobile phone application, StressSense. It runs on Android phones and detects stress using Gaussian Mixture Models (GMMs). The voice features used are pitch along with standard deviation and value intervals of pitch, volume, speaking rate, MFCC and Teager based coefficients. During the development phase, a universal stress model is created, along with several personalised models and an adaptive model. The highest reported accuracy of the universal model is 71.3% when voice is acquired indoor, in supervised conditions using an array of microphones and 66.5% when voice is acquired outside, in uncontrolled, noisy environments.

EmotionSense [13] is another mobile phone app that aims to recognize users' emotions. It also uses GMMs, using the HTK [14] implementation. An advantage of this app is that it also contains a module that handles speaker recognition, so it only checks emotions in the voice of the preregistered user. The mental states recognition performances are not very relevant for the current work, as the system is designed to recognise many emotions: happy, sad, scared, angry and neutral, each containing multiple derived emotions. The highest reported classification accuracy is around 75% for the primary classes and less than 60% for the derived emotions. It is interesting to observe that the system accuracy varies with the length of the classified segment of voice. Accuracy improves with the increase in size of the analysed vocal signal, the improvement becoming insignificant once the length of the signal reaches 10 seconds.

III. VOICE FEATURES

This section describes the voice features that are used in the development of the described app.

A. Mel Frequency Cepstrum Coefficients

The MFCCs describe the signal in terms of power spectrum. The MFCCs are computed using a triangular filterbank, whose values are computed on the mel scale, applied on the spectrum computed with a Discrete Fourier Transform (DFT).

B. The delta and delta-delta coefficients

The delta and the delta-delta coefficients may be computed based on the MFCC coefficients as the first and, respectively, the second derivative of the MFCCs. While the MFCCs describe the signal in terms of power spectrum, the delta and delta-delta coefficients describe the dynamics of the signal.

C. Pitch

Voice pitch is a subjective feature that correlates with the way a voice is perceived as higher or lower. For musical sounds, which are clear and stable, pitch is represented by the fundamental frequency, F_0, which is an inherent property of periodic signals. In the case of the voice signal, which is composed by multiple frequencies and constantly changes in time, pitch cannot be determined by a specific formula, but it can be estimated through the use of different algorithms called pitch estimators. As with musical sounds, voice pitch estimators approximate the fundamental frequency.

IV. BUILDING THE STRESS CLASSIFIER

A. Acquiring experimental data

Considering the fact that the mobile app must be able to identify the stress state of different people, a universal stress model was created. The dataset used for building the model was acquired during a stress inducing experiment session. The work presented in this section is based on previous work presented in [15], meaning it uses a subset of the same experimental dataset. The experiment is based on the Trier Social Stress Test (TSST) [16], containing a public speaking task and a cognitive task, preceded and followed by two neutral tasks. The experiment sessions were performed with the help of 18 student volunteers, aged 18 to 39. More details about the experiment sessions may be found in [15]. Although multiple physiological and behavioral signals were recorded during the experimental sessions, the current work will only use the voice signal, stored as *.wav* files by a device worn around the neck by each participant, the Sociometric Badge [17].

B. Building the dataset

Due to the fact that each *.wav* file consists of more than 10 minutes of voice recording, the processing was done on a PC application written in Java. The desktop application follows the same steps and uses exactly the same libraries and functions for voice processing as the mobile app that will be presented later.

The voice signal is divided in 20 ms frames. A voice activity detector is used to determine if each of the frames is voiced or unvoiced. If the frame is voiced, the following voice features are extracted: first 13 MFCCs, the corresponding 13 delta and 13 delta-delta coefficients and the voice pitch. A feature vector is constructed in the following way:

$$x^t = \left\{ \mathrm{mfcc}_t^i, \delta_t^i, \delta - \delta_t^i, pitch_t, i = 0 \ldots 12 \right\}, \quad (1)$$

consisting of 40 features for each moment of time.

For each feature vector, a label is added, based on the type of activity that each participant was performing at the corresponding time t of the recording: *stressed* if the participant was performing a stressful activity and *neutral*, if the participant was performing a neutral task. The corresponding numeric label is $y_t \epsilon\, Y = \{-1, 1\}$, where -1 corresponds to the neutral state and 1 to stressed.

This is performed on each of the 18 *.wav* files. The complete dataset is constructed by merging the dataset built for each participant: $D = \bigcup_{k=1}^{18} D_k$, where $D_k = \{(x_t, y_t), k = 1 \ldots 18\}$. The size of the complete dataset is $|D| = 1.378.172$, where 792.700 examples correspond to the stressed state and 585.512 correspond to the neutral state.

C. Performing the classification

The stress detection algorithm to be used is developed using the Support Vector Machines (SVM) algorithm [18]. SVM is known as a maximum-margin classifier, meaning that given a dataset of the form:

$$D = \{(x_i, y_i), x_i \,\epsilon\, \mathbb{R}^n, y_i \,\epsilon\, \{-1,1\}\}, \quad (2)$$

where x_i represents the feature vector at the moment of time i and y_i represents the corresponding label, the algorithm tries to find the hyperplane given by the equation $< w \cdot x_i > +b = 0$ that is at maximum distance between the two classes, by finding a solution to the optimization problem:

$$\min_{(w,b,\xi)} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \right\}, \quad (3)$$

subject to the condition:

$$y_i(< w \cdot x_i > -b) \geq 1, \quad (4)$$

where $\xi_i \geq 0$ represent the slack variables that represent the misclassification error of vector x_i and C is the cost parameter. In equation (4), instead of the dot product, $< w \cdot x_i >$, there is usually a kernel function that has the role of mapping the data into a higher dimension.

For performing the classification, libSVM [19] was used, a software library containing applications for: splitting data in subsets, choosing the parameters for a model, performing training on the data, prediction on new data and many other tools. We split the data in two subsets, using 75% for training and 25% for testing. We have used a stratified selection that ensures the same class distribution in each of the data subsets. For the current work, the gaussian kernel was selected. We first performed a grid search to select the best parameters for the value of C – the cost parameter and γ – the Gaussian kernel parameter, so that the parameters are fitted for the testing data. Then the model was trained using the found parameters on the training data. This way a universal stress model was created. The model file contains the kernel type, the model parameters and the support vectors, that help create the boundary hyperplane. Then we measured the performances of the created model on the testing data in terms of accuracy (number of true positives and negatives divided by the number of all the testing examples), precision (number of true positives divided by the sum of true and false positives) and recall (number of true positives divided by the sum of true positives and false negatives). The classification results are shown in Table 1.

TABLE I.
CLASSIFICATION RESULTS OF THE UNIVERSAL STRESS CLASSIFIER

Measure of performance		
Accuracy	Precision	Recall
0.783	0.711	0.871

V. DESIGNING THE TELEMONITORING APPLICATION

In order to monitor the stress state of the users, a client-server architecture was chosen. The client side is represented by a mobile app that runs on Android devices. It communicates with a server that gathers information from multiple clients and centralizes the information in a central database, whose content is made available through a web interface.

A. Developing the Client app

The mobile app was developed to run on Android devices, so it was developed in Java using Android Studio. For voice processing, an open source Java Library was used, called TarsosDSP [20]. It is designed to continuously monitor the stress state without the users' interaction. The application is programmed to run in the background at predetermined time intervals between predetermined hours during the day. Each time the app runs, a 10 second audio signal is acquired using the device's microphone. The signal is segmented in frames of 20 ms. If voice is determined, then the voice features are extracted and saved in a file in the format required by libSVM. The universal stress model is saved as a resource of the app and used to determine if the acquired voice corresponds to a stressed or neutral state. One of the following three possible results is shown to the user: Stressed/ Unstressed/ No vocal segment was identified. If less than 50% of the frames are voices, the result is *No vocal segment was identified.* Otherwise, the result is computed after a comparison between the number of frames labeled as *Stressed* and the number of frames labeled as *Unstressed*. The result is saved in a local database implemented in SQLite along with the time of the recording. The determined state, along with the time of the recording and the user name are packed in JSON format and sent to the server through *http* POST requests for archiving.

The app consists of 5 activities (an Android term similar to GUI). The determined stress states may be visualized by the user in a tabular form in the main activity of the app, called *History*. The user may select the ordering of the data: sorting data by time or by detected stress state. Another activity, *View Preferences*, allows the user to select the time interval for which he wants to visualize the recorded states. The monitoring time interval and the start and end hour may be changed from the application menu, in the *Monitoring preferences* activity. In the same activity, the user may change his username and also the address of the server. Another activity, *Detect state*, allows the user to determine his stress state at any given time, by recording a 10 second voice segment. The determined state is shown to the user on the same activity and also stored in the local database. The app contains an activity called *About* that presents information about the app and how to use it.

B. Developing the Server Application

The server's main purpose is to centralize the information received from the Android app users and present and manipulate this information according to the user's preferences.

The server side allows for the mobile app to be used in different scenarios. For example, the mobile app may be used to monitor the state of people that need assistance in their daily life, like the elders or people struggling with a form of disability. In this case, the users of the server application may be formal or informal caregivers that may in this way detect the activities that are stressful for assisted persons and come with solutions for specific problems. Another use case scenario is the case of a psychiatrist monitoring his patients. The patients may not remember every stressful situation or they may not evaluate the situations correctly. By using the designed telemonitoring application, the psychiatrist has constant access to precise information about his patients' mental states. Another use case scenario involves monitoring the mental state of the employees working in different areas. This would allow the managers to identify stressful situations and to better allocate human resources.

The server was developed using Apache Tomcat, using Eclipse Enterprise Edition as the developing environment. It is developed as a REST server. The database was built in PostgreSQL using Hibernate for data persistency support. For consulting the information from the database, a web interface was developed using Java Server Pages (JSP) that allows the integration of Java code in html code. The web page is available on any Internet connected device through a web browser and it presents the information from the database in tabular form. It allows sorting of the data by any of the table head cells in ascending/descending order and it also allows filtering the data to be shown by searching for the recordings to be shown through a text input box.

VI. Conclusions and Future work

The paper describes the steps involved in designing and implementing a mobile app for stress monitoring using voice features. The app monitors the stress states by using only voice extracted features with an accuracy of 78%. The mobile app is easy to use and completely non-invasive, not requiring any external sensors or devices. It should be noted that voice is stored only in temporary buffers and that only the detected states are stored.

The current work will be improved by adding a speaker recognition module. This will impose processing and time resources penalties, but will assure the fact that the detected stress states correspond to the targeted user. This is a necessary step that will increase the reliability of the system.

The server application allows for the mobile app to be used for telemonitoring in different scenarios like: monitoring the state of people that need assistance in their daily lives, to detect the activities that are stressful for them; monitoring patients by a psychiatrist, to allow the psychiatrist to have a better view of the mental states of his patient; monitoring the employs in different working areas to be able to better allocate human resources; etc.

For the moment, all the processing is done on the client side, to minimize cellular data usage. It is more efficient to perform the computationally expensive phases on the server, as this runs on a PC that usually has more computation power even than the most modern smartphones. A possible improvement is allowing the user to choose whether to perform the computationally expensive processing on the server, with the price of using his data plan, or perform the computation on his device. The server side application for voice features extraction and stress classification is already created as it was needed to create the universal stress model.

Currently, the mobile app only keeps a log of the detected stress states of the user, without providing any feedback. In future developments, the app may comprise a module that allows communicating with the user. For example, when detecting the stress state, the app may suggest the user some relaxing music, present points of interests in the area that may held pleasant evens, recommend an outdoor activity or other relaxing activities. These messages should be correlated with the frequency of detected stressful moments.

Another possible improvement is the implementation of an additional view of the detected states in graphical form or by offering synthetized information, for example: the number of detected stressful moments in a day or in a selected time interval, reports about the most stressful periods, etc.

On the server side, further work may be done to secure the connection between the server and clients, by accepting only secure connections and by implementing the exchange of automatically generated security tokens, like RSA SecurID.

Acknowledgment

The work has been partially funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132397.

References

[1] P. A. Thoits, “Stress and health: major findings and policy implications.,” *J. Health Soc. Behav.*, vol. 51 Suppl, pp. S41–S53, 2010.

[2] A. Trusina, “Stress induced telomere shortening: longer life with less mutations?,” *BMC Syst. Biol.*, vol. 8, p. 27, 2014.

[3] E. S. Epel, E. H. Blackburn, J. Lin, F. S. Dhabhar, N. E. Adler, J. D. Morrow, and R. M. Cawthon, “Accelerated telomere shortening in response to life stress.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 101, no. 49, pp. 17312–17315, 2004.

[4] J. Choi and R. Gutierrez-Osuna, “Estimating mental stress using a wearable cardio-respiratory sensor,” *Proc. IEEE Sensors*, pp. 150–154, 2010.

[5] Y. Shi, M. Nguyen, P. Blitz, and B. French, “Personalized stress detection from physiological measurements,” *Int. Symp. Qual. Life Technol.*, pp. 28–29, 2010.

[6] H. Costin, C. Rotariu, and A. Păsărică, “Mental stress detection using heart rate variability and morphologic variability of ECG signals,” *Proc. 2012 Int. Conf. Expo. Electr. Power Eng. (EPE 2012)*, pp. 591–596, 2012.

[7] A. Sano and R. W. Picard, “Stress Recognition using Wearable Sensors and Mobile Phones,” *Hum. Assoc. Conf. Affect. Comput. Intell. Interact. Stress*, pp. 671–676, 2013.

[8] K. Chang, D. Fisher, and J. Canny, “Ammon: A speech analysis library for analyzing affect, stress, and mental health on mobile phones,” *Proc. PhoneSense*, 2011.

[9] ETSI, “ES 202 212 Extended advanced front-end feature ex- traction algorithm v1.1.4. Technical report,” pp. 1–93, 2005.

[10] F. Eyben, F. Weninger, F. Gross, and B. Schuller, “openSMILE - The Munich Versatile and Fast Open-Source Audio Feature Extractor,” *Proc. ACM Multimed.*, pp. 835–838, 2013.

[11] X. Zuo and P. Fung, “A Cross Gender and Cross Lingual Study on Acoustic Features for Stress Recognition in Speech,” *Proc. Int. Congr. Phonetic Scences*, pp. 2336–2339, 2011.

[12] H. Lu, A. T. Campbell, and D. Gatica-perez, “StressSense□: Detecting Stress in Unconstrained Acoustic Environments using Smartphones,” *Ubicomp*, pp. 351–360, 2012.

[13] K. K. Rachuri, C. Mascolo, P. J. Rentfrow, and C. Longworth, “EmotionSense□: A mobile phones based adaptive platform for experimental social psychology research,” *Ubicomp ’10 Proc. 12th ACM Int. Conf. Ubiquitous Comput.*, pp. 281–290, 2010.

[14] Cambridge University Speech Group, “Hidden Markov Model Toolkit (HTK).” [Online]. Available: http://htk.eng.cam.ac.uk/. [Accessed: 09-Mar-2013].

[15] V. Sandulescu, S. Andrews, D. Ellis, N. Bellotto, and O. M. Mozos, “Stress Detection Using Wearable Physiological Sensors,” in *Artificial Computation in Biology and Medicine*, Springer International Publishing, 2015, pp. 526–532.

[16] R. Miller and C. Krischbaum, “Trier Social Stress Test,” in *Encyclopedia of Behavioral Medicine*, Springer New York, 2013, pp. 2005–2008.

[17] D. Olguín Olguín, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland, “Sensible organizations: Technology and methodology for automatically measuring organizational behavior,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 1, pp. 43–55, 2009.

[18] C.□; Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[19] C.-C. Chang and C.-J. Lin, “LIBSVM□: a library for support vector machines.” ACM Transactions on Intelligent Systems and Technology, pp. 2:27:1–27:27, 2011.

[20] J. Six, O. Cornelis, and M. Leman, “TarsosDSP, a Real-Time Audio Processing Framework in Java,” *Proceedings of the 53rd AES Conference (AES 53rd)*, 2014