

Zeno the Voice Recogniser

Practical report

Teachers: Prof. Pedro Vieira e Prof. Hugo Gamboa

Realised by: Diogo Durão n° 55739 e Mónica Dyreby n° 55808

Dep. de Física, SST-UNL

December 2022

Contents

1	Introduction	1
2	Dataset	1
3	Extracted features and libraries	1
4	Classifier choice and statistics	2
5	Real time Classification and instructions	3
5.1	The process from website launch to classification:	4
5.1.1	Data-set and training	4
5.1.2	Website inputs	4
5.1.3	Recording	4
5.1.4	Classification	5
5.1.5	Extras	5
6	Limitations and conclusion	5
7	Attachments	6

List of Figures

1	Orange structure	2
2	Confusion matrix with the SVM classifier with only two features and a random test sample of 33% (see table 1 for more information)	2
3	Confusion matrix with the SVM classifier with only three features and a random test sample of 33% (see table 2 for more information)	2
4	Two-dimensional plot of the first two features. There is some confusion between Tiago and Mónica. The extra red dot illustrates audio recorded on a different day, by Mónica, under different circumstances than the database. This red dot exists in between the Tiago and Mónica data.	3
5	Three-dimensional plot of the three features. The red dot in the figure originates from the same audio as the one mentioned in Fig. 4. Now the difference between Tiago and Mónica is clear and the red dot audio would clearly be classified as Mónica	3
6	Screenshot from the website requesting data. In this case, the forms are already filled out.	4
7	Audio file plot with classification (Diogo) that is displed on the website once the classification is finished.	5
8	Original audio file plot, trimmed and cleaned	6
9	Evaluation results, from Orange DM, for different classifiers with the top two features and a random test sample of 33%	6
10	Evaluation results, from Orange DM, for different classifiers with the top three features and a random test sample of 33%	6
11	Screenshot from the website with Time equal to zero.	7
12	Screenshot from the website with a Host Ip that does not match the one in the LANmic application.	7

List of Tables

1	Classification report for the SVM classifier with two features and 33% of the data set used as a test	6
2	Classification report for the SVM classifier with three features and 33% of the data set used as a test	6
3	Average maximum and average minimum audio intensity of the data-set audio files for the different classifications and for Silence. Since Silence was not to a part of the database, 20 audio files with 10s each of background noise were recorded. Audio files with average intensities bellow 0.02 are considered Silence.	7

1 Introduction

The motivation for this project was to create a hands-free unlock method for a mobile phone. This should be done with a specific set of sentences, that the device would recognise as being said by the owner of the device.

With this in mind, this project used: the LANmic mobile application to stream sound to a computer and a Raspberry Pi to analyse the audio file and host a website, which will classify the speaker as Diogo, Tiago or Mónica and hypothetically unlock a phone.

The Raspberry Pi was used to host a website, via a socket server, that requests inputs (LANmic host id, sample rate and acquisition time in seconds) and establishes a connection between the Raspberry Pi and the LANmic application once these have been filled out. If activated, LANmic will send audio data back to the Raspberry Pi and an audio file is created, "cleaned" and its features are extracted.

These extracted features will be placed in a SVM machine learning classifier and referenced against a database, returning with one of these three classifications: Diogo, Mónica or Tiago. If the voice input intensity is too low, the program instead classifies it as Silence without needing to use the SVM classifier. Essentially, the program in the Raspberry Pi will determine if the audio file was recorded by Diogo, Mónica or Tiago, or no one. To ensure the number of features is small and to make sure a simple classifier can be used, the audio file features in the database all originate from the same sentences being said by these three people. Thus the file recorded via the website and LANmic should also feature the same set of sentences:

"Hey Zeno, sou eu. A password é 'So long and thanks for the fish'. Por favor, desbloqueia-me o telemóvel."

2 Dataset

LANmic is a mobile phone application that captures sound from the microphone and sends the data via streaming to the computer. This app was used for the acquisition of this project's voice data. In the application, a 44100 S/s was selected (the max rate of the app), the encoder as WAVE, and sent via HTTP protocol. To make the connection between the computer and LANmic and save the respective data, Prof. Pedro Viera's program [1]"lanmic.py"¹, was utilised while both the computer and the phone were connected to the same WiFi network.

With this setup ready, each of the three participants recorded their voice saying the following line: "Hey Zeno, sou eu. A password é "So long and thanks for the fish". Por favor, desbloqueia-me o telemóvel." until each had 50 .wav files catalogued respectively with the name of its class, that is, the name of the participant (Diogo, Mónica or Tiago). In total 150 .wav files were recorded and saved. Each recording was done in a ten-second window.

Afterwards, all of these files were treated in the following way: the silent parts in the audio are removed via a high-pass filter (that cuts off all the sound 20 dB below the most intense signal) in two processes called "audio trimming" (cuts the silence from the beginning and end of the audio) and "audio splitting" (cuts the silent parts from the middle portions of the audio). This was done in order to prevent the characteristics (mfcc, delta and delta delta) of background noise, present in speaking pauses, to be associated with the audio files. Fig. 8 in **Attachments** illustrates the effects of trimming and cleaning.

With the audio files treated, the spectral audio features (the MFC coefficients or MFCC) can be extracted from these audio files with the appropriate frame length and hop length (this will be clarified in section **Extracted features and libraries**). The first and second derivative coefficients of the MFCCs were also extracted. Since each coefficient is actually a coefficient array (each of these coefficients varies with the recording time), the mean value of each array was the actual feature selected for machine learning, in order to simplify the classification model. Finally, these features are introduced into a .csv table where each line represents a recording, with its respective class and extracted parameters.²

3 Extracted features and libraries

As it was mentioned before, the features extracted are the Mel-Frequency Cepstrum Coefficients (MFCC), which are one of the most used features for human speech analysis nowadays. These coefficients are derived in the following way[2][3]:

1. Frame the signal into blocks of 20 to 40 ms.

¹While this program was created by Prof. Pedro Viera, it was slightly tweaked to fit this project's needs

²The audio files were cleaned and their features were extracted with a program created to remove these features and turn them into a csv file for all the 150 audio files recorded. This program is not included in the folder that launches the website since it is not necessary to do so. It only creates the dataset and once this is created the program is no longer needed.

2. Perform a Fourier transform on the framed signal and calculate the powers of the spectrum
3. Map these powers to the mel scale, which represents the interpretation of the audio listener on pitch scaling. Frequency can be converted to the mel scale with the following formula:

$$m = 1127 \ln(1 + \frac{f}{700}) \quad (1)$$

Additionally, triangle overlapping filters are also used for mapping, by multiplying these with the spectrum power formerly calculated.

4. Compute the logarithm of the power
5. Compute the discrete cosine transformation of the filterbank energies to get the MFCC coefficients.

For step 1, a frame length of 20 ms was chosen, and a hop length of 10 ms. It is necessary to have overlapping frame data with a selected hop size to avoid a continuity loss between frames which could misrepresent the audio frequencies.

While the MFCCs array sizes depend on the frame and hop length, the number of MFCCs, n , should be chosen according to the sample rate, sr . There is the following rule of thumb to choose n , with sr in KHz:

$$n = \frac{3 * sr}{4} + 1 \quad (2)$$

The sample rate of the stereo audio recorded was 44.100 KHz, therefore in mono audio is 22.050 KHz, which gives $n = 18$ coefficients (rounded up).

Other parameters were extracted for this dataset: the delta and the delta delta coefficients, which characterise the first and second-time derivative of the MFCCs. There's also 18 coefficients of each type, therefore the training dataset as 54 features. As seen in 4, only MFCCs were used for the classification.

All of these steps were done with the audio analysis library **LibROSA**. This library has the function *librosa.feature.mfcc*, which does step 1-5 single-handily. It also offers other useful sound processing features, like wave signal visualisation (which will be used in the website).

Another important utilised library is **sklearn**, which is a library built for machine learning. This library was used to build the classification model, evaluate the performance and visualise the results.

For the communication between the program and the HTML page, the micro web framework **Flask** was utilised, which is ideal for developer servers and debugging on Python.

The following libraries were also used: time, matplotlib, csv, os, gob, numpy, sys, and statistics.

4 Classifier choice and statistics

Once the data-set was finalised, the resulting csv file was imported onto the Orange Data Mining application, and the structure in Fig. 1 was created.

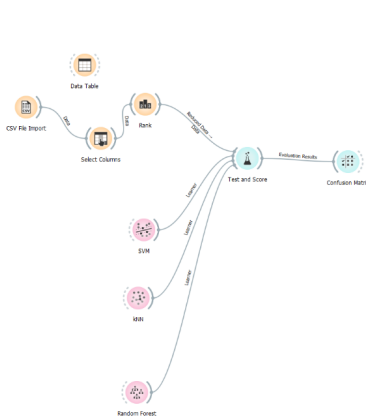


Figure 1: Orange structure

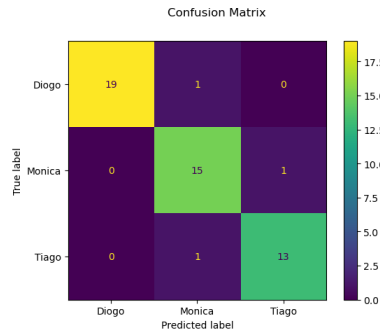


Figure 2: Confusion matrix with the SVM classifier with only two features and a random test sample of 33% (see table 1 for more information)

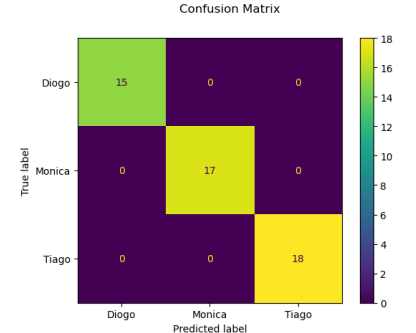


Figure 3: Confusion matrix with the SVM classifier with only three features and a random test sample of 33% (see table 2 for more information)

From here, the features in the database were ranked and the top three were selected (mfcc_2, mfcc_6, mfcc_11), as well as the classifier which provided the most accuracy which was the SVM classifier (see Figs. 9 and 10 in **Attachments**).

Initially, only two features were selected: mfcc_2 and mfcc_6, since they provided an adequate confusion matrix, with very high accuracy (see figure 2). However, once this model was used to classify audio files recorded under different circumstances than those in the database (like background noise, packet losses³ and differences in inflexion), the accuracy decreased. This is illustrated in Fig. 4. Thus a third feature was added to the SVM classifier: mfcc_11. The issues witnessed in Fig. 4 are resolved with the third feature as is clear by Fig. 5. The confusion matrix with the three features is in Fig. 3.

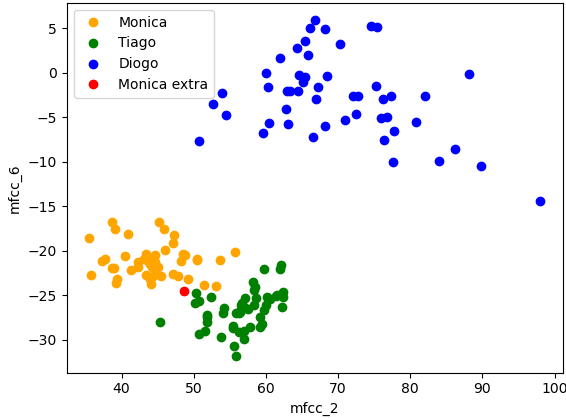


Figure 4: Two-dimensional plot of the first two features. There is some confusion between Tiago and Mónica. The extra red dot illustrates audio recorded on a different day, by Mónica, under different circumstances than the database. This red dot exists in between the Tiago and Mónica data.

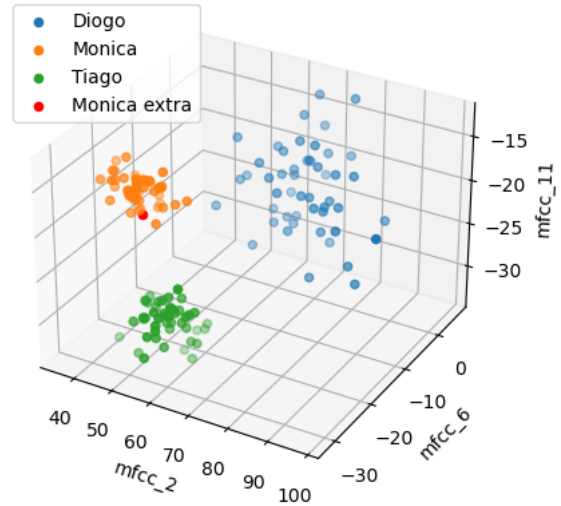


Figure 5: Three-dimensional plot of the three features. The red dot in the figure originates from the same audio as the one mentioned in Fig. 4. Now the difference between Tiago and Mónica is clear and the red dot audio would clearly be classified as Mónica

While it might seem, from Fig. 3, that the accuracy is 100%, when this model is applied to a real-time recording, the accuracy decreased, since this model is not taking into audio recorded under new circumstances. In particular, there was still some confusion between Tiago and Mónica even with the third feature.

Tab. 1 and 2 in **Attachments** show the difference in precision between the two and three feature models. Clearly, three is more precise. These were obtained using the `metrics.classification_report()` function imported from `sklearn`.

5 Real time Classification and instructions

As was mentioned before, in this project a website is hosted by a socket server in a Raspberry Pi and audio is recorded via the LANmic application and then analysed in the Raspberry Pi to determine the identity of the audio creator. For this to happen in real-time a folder was created in the Raspberry Pi with the following files:

- **static folder:** Folder with all the images (.png and .jpeg) featured in the website and the `cute.css` that stores style information for the web page;
- **templates folder:** Folder with the `index.html` that structures the website and sets up communication with the Raspberry PI and the user interface. This HTML is based on the one made available in Lab 6;
- **app.py:** Flask web app that launches a built-in server for the website in Raspberry Pi. It establishes a connection between `feature_extract.py`, `lanmic.py`, `total_final_dataset.csv` and `index.html`;
- **lanmic.py:** Python program that establishes a connection with the LANmic application and records the audio, creating a wave file named "predict_sample.wav" in the folder;

³Will be discussed in **Limitations and conclusion**

- **feature_extract**: Python program that analyses the average intensity of the "predict_sample.wav". If it is above a 0.02 threshold⁴, it extracts the three features from the "predict_sample.wav" and outputs a classification for the audio: Diogo, Mónica or Tiago. If not, it classifies it as Silence;
- **total_final_dataset.csv**: CSV file with the dataset created, for more information see section **Dataset** ;

5.1 The process from website launch to classification:

5.1.1 Data-set and training

Before the website is live, via the flask application app.py, the csv file is read, and the data labelled mfcc_2, mfcc_6 and mfcc_11 is loaded into an array composed of 150 arrays with a [mfcc_2, mfcc_6, mfcc_11] structure. The first 50 have information regarding the Diogo classification, the next 50 regarding Monica and the last 50 regarding Tiago. The final array would look something like this:

[[mfcc_2, mfcc_6, mfcc_11], [mfcc_2, mfcc_6, mfcc_11], ... , [mfcc_2, mfcc_6, mfcc_11]]

The labels (Diogo, Monica and Tiago) are also loaded into an array with a 150 length. The structure would look like this:

[Diogo, Diogo, ... , Monica, ... , Tiago]

These two arrays were then loaded onto the SVM classifier, imported from the sklearn library, to train it.

5.1.2 Website inputs

After the training, the website becomes accessible and it requests three inputs, as is seen in Fig. 6. These inputs are: Host IP from the LANmic application, the Sample Rate⁵ in the LANmic application, the acquisition time in seconds (which determines the length of the recorded audio) and the Raspberry Pi IP.

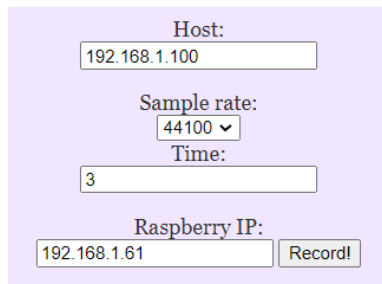


Figure 6: Screenshot from the website requesting data. In this case, the forms are already filled out.

Once these have been filled out and the "Record!" button is pressed the socket is initialised and a connection request is sent by the client-side socket object and received by the server-side socket. If the request is successful, it will send the Host, Sample rate and Time inputs to the server-side socket to be analysed in the app.py program.

5.1.3 Recording

The three inputs received in app.py will then be used as parameters for the record function (imported from the lanmic.py program) and an audio file named "predict_sample.wav" is created. The record function returns the name of the wave file. To increase the website's ergonomics and to counter audio information loss due to potential delays in the LANmic-Raspberry Pi connection, a message is emitted and displayed in the website. This message is sent as soon as the LANmic-Raspberry Pi connection is established and not as soon as the "Record!" button is pressed. This message reads: "Sound being acquired ...".

⁴This value was determined by the analysis of the maximum and minimum audio intensity of the audio files that created the data-set (see Tab.3 in **Attachments** for more information).

⁵While there is a dropdown menu with many different sample rate options that match the ones available in LANmic, only the 44100 Samples/s should be used, since the csv database used this sample rate.

5.1.4 Classification

The audio file name is then introduced as a parameter in the extract function (imported from the feature_extract.py program) as well as the trained model created in the beginning. As was mentioned above this, this function returns the classification: Diogo, Monica, Tiago or Silence. A .png file is also created that displays the sound wave created in the record function, and it also relays the classification (see Fig.7). Both the image

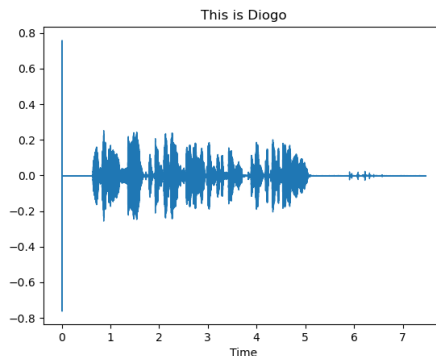


Figure 7: Audio file plot with classification (Diogo) that is dispiled on the website once the classification is finished.

and one of the following messages: "Hello, Diogo. I have unlocked your phone!" or "I didn't quite catch that. I only heard silence. Please try again." are displayed on the website, thus effectively informing the website user of the classification.

5.1.5 Extras

Focusing a little on website ergonomics, some extra features were added:

- Error message and broken clock .jpeg image displayed in the website when the input Time is zero or less (see Fig. 11 in **Attachments**);
- A try-except block in the record function that tries to create the LANmic-Raspberry Pi connection. If this connection fails, it displays an error message on the website. This connection could fail due to: wrong input in the Host form, the LANmic application being off, and LANmic and Raspberry using different WiFi (see Fig. 12 in **Attachments**);
- The .png files created after each sound recording and displayed on the website are labelled with the time() function. This guarantees that the name of the file is always different and the images are replaced in the website, once a new recording is created. To avoid filling up the static folder with all the audio file plots, after each recording all .png files are deleted from said folder.

6 Limitations and conclusion

While this system can distinguish three people by their voice or determine if the wave file created is Silence it has some limitations. For instance: the phone where the LANmic application is being run, needs to be the same as the one used to create the data-set, otherwise, the classification will fail. In other words, this website is specific to one mobile device!

Moreover, as it is possible to see in Fig. 7, all audio files created come with a spike at the beginning. This could be due to a programming lapse or some other issue with LanMIC. This doesn't generate a big error for the classification, but makes the generated wavesignal plot with a bigger y axis.

Another limitation is the Sample rate. As was mentioned above, the selected sample rate in the website should be 44100S/s since it is the sample rate used to create the data-set. A failure to do so could compromise feature extraction and classification.

Finally, this system requires a great WiFi connection. If not, there could be some packet losses when the audio is being recorded, which could lead to errors in classification.

Nonetheless, it was possible to successfully create an application that classifies the voice of three different people, which could be a very hard task if it was not for the resorting of machine learning techniques.

7 Attachments

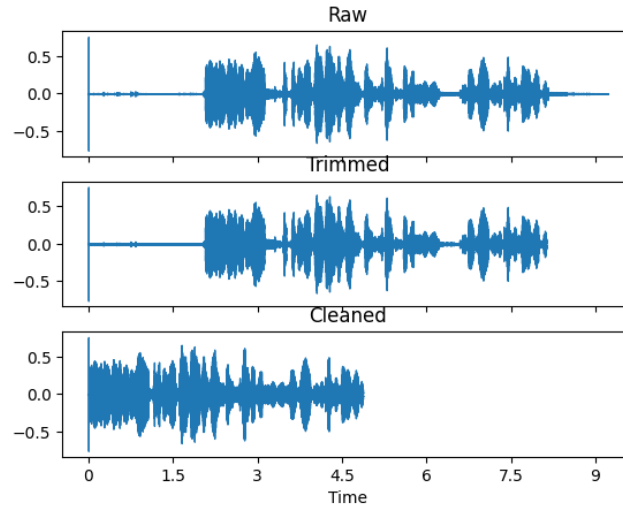


Figure 8: Original audio file plot, trimmed and cleaned

Table 1: Classification report for the SVM classifier with two features and 33% of the data set used as a test

	precision	recall	f1-score	support
Diogo	1	0.95	0.97	20
Monica	0.88	0.94	0.91	16
Tiago	0.93	0.93	0.93	14
Accuracy			0.94	50

Table 2: Classification report for the SVM classifier with three features and 33% of the data set used as a test

	precision	recall	f1-score	support
Diogo	1	1	1	15
Monica	1	1	1	17
Tiago	1	1	1	18
Accuracy			1	50

Evaluation Results					
Model	AUC	CA	F1	Precision	Recall
kNN	0.995	0.968	0.968	0.969	0.968
SVM	0.998	0.974	0.974	0.975	0.974
Random Forest	0.995	0.960	0.960	0.961	0.960

Figure 9: Evaluation results, from Orange DM, for different classifiers with the top two features and a random test sample of 33%

Evaluation Results					
Model	AUC	CA	F1	Precision	Recall
kNN	1.000	1.000	1.000	1.000	1.000
SVM	1.000	1.000	1.000	1.000	1.000
Random Forest	0.999	0.980	0.980	0.981	0.980

Figure 10: Evaluation results, from Orange DM, for different classifiers with the top three features and a random test sample of 33%

Table 3: Average maximum and average minimum audio intensity of the data-set audio files for the different classifications and for Silence. Since Silence was not to a part of the database, 20 audio files with 10s each of background noise were recorded. Audio files with average intensities bellow 0.02 are considered Silence.

1	Max	Min
Monica	0.114	0.084
Diogo	0.059	0.026
Tiago	0.098	0.650
Silence	0.022	0.005

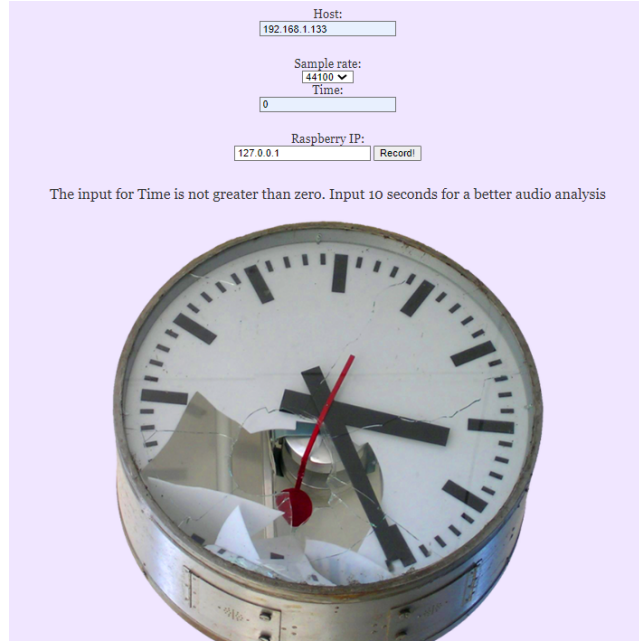


Figure 11: Screenshot from the website with Time equal to zero.

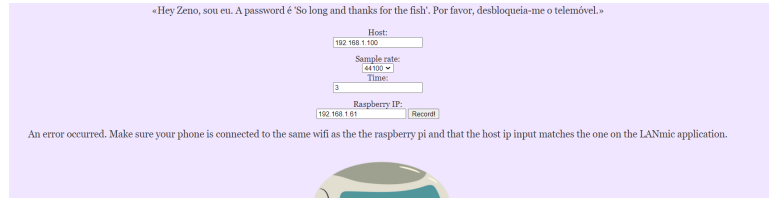


Figure 12: Screenshot from the website with a Host Ip that does not match the one in the LANmic application.

References

- [1] P. Viera. *app-pv-lanmic3.py*. 2022. URL: <https://github.com/hgamboa/nova-adv-inst/commit/590e84e57e8e5c447bb4289cf17b7affd1bd1f9f>.
- [2] Abhishek Manoj Sharma. “Speaker Recognition Using Machine Learning Techniques”. San Jose State University, May 2019. DOI: 10.31979/etd.fhhr-49pm. URL: https://scholarworks.sjsu.edu/etd_projects/685.
- [3] Md Sahidullah and Goutam Saha. “Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition”. In: *Speech Communication* 54 (4 May 2012), pp. 543–565. ISSN: 01676393. DOI: 10.1016/j.specom.2011.11.004.