# Speech & Speaker Recognition for Romanian Language

**Vezeteu Eugeniu**
Faculty of Mathematics and Informatics
University “Ovidius” of Constanta
Mamaia 124 Constanta, Romania
vezeteu.eugeniu@yahoo.com

**Sburlan Dragos**
Faculty of Mathematics and Informatics
University “Ovidius” of Constanta
Mamaia 124 Constanta, Romania
dsburlan@univ-ovidius.ro

**Pelican Elena**
Faculty of Mathematics and Informatics
University “Ovidius” of Constanta
Mamaia 124 Constanta, Romania
epelican@univ-ovidius.ro

## ABSTRACT

The present paper illustrates the main methods that can be employed to build a speech and speaker recognition system for Romanian language. To this aim, we start by presenting the classical approach of extracting the Mell Frequency Cepstral Coefficients features from a dataset of speech signals (which represents some words/phrases in Romanian language). The recognition is done either by using Dynamic Time Warping (DTW) or by training an Convolutional Neural Network. A comparison between these models is presented and commented.

Once such a system is developed, we proceed further by implementing an application that listens and executes some predefined commands. In our setup, the system performs two main tasks: it recognizes the user by his voice and executes a task corresponding to the vocal command.

Source code can be downloaded at: click to download

**Author Keywords**
Sound processing, feature extraction, pattern recognition.

**ACM Classification Keywords**
Numerical Algorithms and Problems(*fast Fourier transform*), Natural Language Processing (*Speech recognition and synthesis*), PATTERN RECOGNTION (*Neural nets, Dynamic Time Warping*).

## STATE OF THE ART

Nowadays, most of speech recognition systems perform short time spectral analysis in order to obtain the Mell Frequency Cepstral Coefficients (MFCC) as features. For speech and speaker recognition, the feature extraction module is the same. For speaker modeling, Hidden Markov Models (HMM) can be used [15][19], with Vector Quantization technique[12]. In this paper Speaker recognition was performed by implementing the Nearest Neighbor algorithm. In proceeding [23] speech recognition for Romanian language is performed by implementing HMM in spoken dialogue systems. Other approaches to this topic are hosted by Google (i.e. *Google translate*), which offers the final version of the software or imported as a library, but does not allow access to see the source code of the application.

## INTRODUCTION

Automatic natural language processing is a wide subject at the border between scientific fields such as formal language theory, statistics, artificial intelligence, signal processing, and linguistics and has had recently a steady development due to the technological progress.

An implicit application of this field concerns the human interactions with a computer using the natural language as a means of communication (regarded as a tool used by people to express their thoughts and make themselves understood). In this respect, voice interaction represents a complex task for an automated system as it generally assumes solving some fundamental problems:

- recognizing different users by their voices;
- recognizing the words and sentences spoken by the user;
- understanding the semantics of the transmitted message;
- developing an answer that is directly and accurately related to the transmitted information (by extracting and processing the meaningful information out of the message);
- translating the answer into a voice form (that is human understandable).

As the natural language represents the best way of communication for humans, we aim to develop a system from scratch in which the human-computer interaction is achieved through the use of speech. Consequently, in order to create such a system we firstly recorded a dataset of speech samples. We considered a set of words and for each one of them we recorded 5 different speakers for 100 times, the goal was to capture a wide variety of word pronunciation. We recognize that this is one of the limitations, at the moment the dataset is made up of only 700 words. There is no free dataset available, so we had to register one.

Romanian speech recognition was done in two ways. The first one employs the Dynamic Time Warping (DTW) algorithm and the second one is using Convolutional Neural Network (CNN). For both methods we have used as input some features extracted from the dataset - they facilitate the subsequent learning methods.

Feature extraction is done by performing the following steps:

1) silence removal (a step necessary to determine the words boundaries in the input sound);

2) framing (a step that splits the sound wave into chunks out of which the phonemes will be detected);

3) windowing (a step necessary when one wants to analyze the periodic behavior of the sound wave in a short duration);

4) Fast Fourier Transform (FFT) (a step necessary to map time domain signal into frequency domain)[21];

5) Mell Filter (in order to extract the energy from each frequency band);

6) Discrete Cosine Transform (DCT) (a step necessary to map the signal from frequency domain to time domain).

The result of feature extraction is a *feature vector* which will be used by the pattern recognition system to map the input sound to a word (or a sequence of words) or to recognize the Speaker.

## IMPLEMENTATION

### Elements of acoustic wave theory

A sound wave represents a longitudinal wave; it defines a series of alternative compressions and extensions of the environment. An acoustic wave is characterized by several properties:

The *frequency* of a sound is the number of periods or oscillations that a sound wave takes in the unit of time. The standard unit for frequency measurement is the Hertz defined as 1 Hz = 1 vibration / 1 second. The *volume* of sound wave refers to the sound intensity. Sound intensity measures the energy transferred by the sound wave in the unit of time through the unit area of a surface orthogonal to the direction of the wave propagation. The sound intensity is expressed in decibels (dB). *The amplitude* of the sound wave represents the maximum elongation registered by the points from the elastic medium with respect to the equilibrium position. The amplitude of vibrations is proportional with the intensity of the sound. *Timber* refers to the property of a sound to distinguish from another one produced under identical conditions by different sources; the timber of a voice is given by the superior formats (the spectral peaks of the sound spectrum).

As is stated in [13], sounds are complex combination of vibrations (with different frequencies and amplitudes). For example, in Figure 1 three tones with different characteristics are represented: a) frequency 400 Hz, amplitude 0.2; b) frequency 550 Hz, amplitude 0.4; c) frequency 800 Hz, amplitude 0.3.

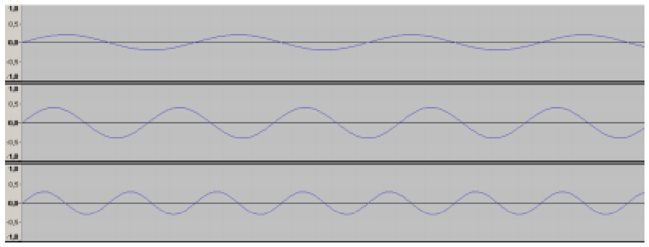


**Figure 1. Three sounds (tones) with different frequencies and amplitudes.**

The composition of these three waves is illustrated in Figure 2. One can notice that the result is no longer a sinusoid (however, the oscillatory behavior is preserved).

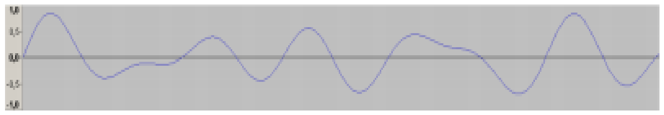


**Figure 2. The composition of the sound waves represented in Figure 1**

The inverse problem (that is, obtaining the frequencies that compose an acoustic signal) is done by using the Discrete Fourier Transform (DFT). Fast Fourier Transform represents a fast algorithm for computing DFT**.**

A *sound spectrum* is a representation of a sound in terms of the amount of vibration at each individual frequency. For instance, the sound spectrum for the sound wave given in Figure 2 is presented in Figure 3.

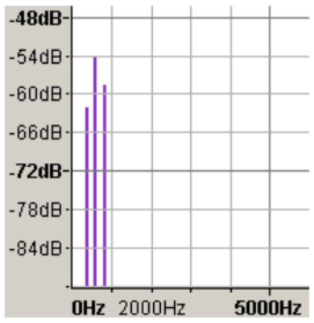


**Figure 3. The sound spectrum for a sound wave**

Once the sound spectrum is obtained, an important task in speech recognition (related to recognition of vowels) is the identification of *formants* – the peeks from the sound spectrum.

### Audio format

The quality of a digital audio recording depends mainly on two factors: the sample rate and the sample format or bit depth. In order to save memory, the items from the dataset were recorded using a fair quality: a 22050 Hz sample rate, 16 bits unsigned sample format, mono channel.

## Feature extraction

The information contained in a speech wave must be extracted as a word sign. Actually one needs to process the short term amplitude spectrum. The feature extraction algorithm is called Mell Frequency Cepstral Coefficients (MFCC), this is a standard method to get the representative points from feature wave and it is widely used for speech and speaker recognition[22]. The motivation of using MFCC is because one wants to mimic the human hearing. Mell scale is linear below 1 kHz for frequency band and it is logarithmic above. The MFCC algorithm consists of several parts:

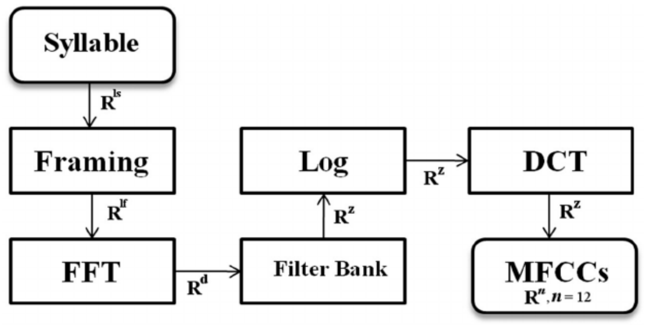


**Figure 4. Feature extraction**

## Pre-Processing

The audio signal is a continue wave which contains not only meaningful information about the speech, but also silence and noise. In order to detect the speech segments, one has to exclude silence and noise from the speech wave. One common algorithm for silence removal is Endpoint detection. It consists of 4 parts [24].

- One has to compute the mean μ and the standard deviation σ for first 200ms. This is done because the first 200ms samples are usually the silence (when people speak, they actually need some time for breathing; these 200ms samples represent the background noise).
- Next, one has to go from the first sample to the end of the wave and check if $|x- \mu|/ \sigma > 3$ or not. If it is greater than 3, then the sample is voiced, else it is unvoiced. The threshold is 3 and can be modified, depending on the quality of the microphone.
- one needs to divide the whole speech wave intro frames of 10ms and mark each frame with 1 if it is voiced and 0 is it is unvoiced.
- One has to go through the entire vector of speech and collect the frames labeled with 1 (they represent the voiced frames).

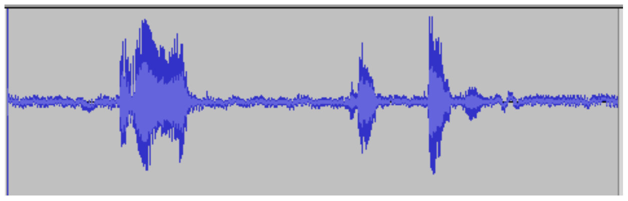


**Figure 5. Audio signal for speech "Buna ce faci" (audio signal is captured with Audacity)**

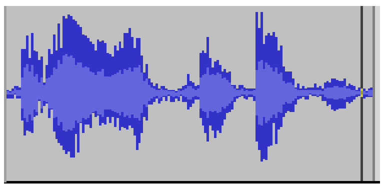


**Figure 6. Result of silence removal**

The algorithm was used for splitting words from a speech wave.

## Framing and windowing

Because Speech is a continuous signal, it means that statistical properties are different during the time. We need to extract feature points from a small window. We assume that statistically the signal does not change much. We shall frame signal into 20-40ms frames. If the frame is too short, we lose the relevant information, if it is longer the signal changes too much, so we can not get relevant points. We use block of frames with 25ms with overlapping 10ms.

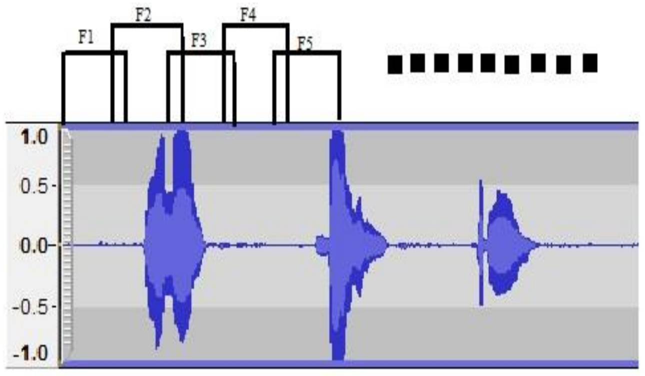


**Figure 7. Framing the audio signal**

After framing the audio signal, we obtained a list of frames. Using Audacity one can zoom in the frame and see how the speech wave is represented (see Figure 9). Extracting the spectral features of each frame was done by using the FFT algorithm (because FFT algorithm has as input a sequence of length of power of 2, in our setup each frame has 512 data points).

In order to eliminate/enhance some spectral features of the audio signal we used the Hamming window. The equation describing the window is:

$$w(n) = \begin{cases} \alpha - \beta\cos(\frac{2\pi n}{N-1}) & \text{if } 0 \le n \le N-1; \\ 0 & \text{otherwise.} \end{cases}$$

where $\alpha = 0.54, \beta = 1 - \alpha = 0.46$

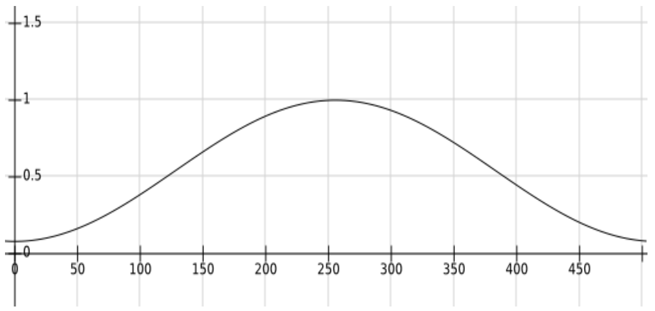


**Figure 8. The Hamming window**

After framing, we multiplied each frame with the Hamming window. If the signal is denoted by *s(n),* where *1≤n ≤N*, N being the length of each frame, then the result is:

$$x(n) = s(n) \cdot w(n)$$

The resulted signal takes the shape of the window and become smoother and flatter.

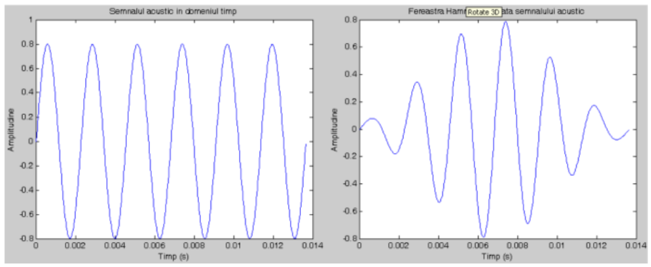


**Figure 9. A sound wave before and after windowing**

One can notice that at both ends of the frame, the amplitude of the signal is reduced; consequently, the periodic „jumps” that appear while computing DFT are more diminished (recall that DFT assumes the existence of a infinite long signal while here we worked with a finite one). It follows that, by using the Hamming window, some artificial peeks in the spectral analysis are removed, while the „useful” ones are preserved.

**Short time energy and Zero crossing rate**

After framing and windowing one needs to remove the silence from each frame. To this aim we computed the *energy* of each frame by the formula:

$$Energy = \sum_{t=t_1}^{t_2} x(t)^2$$

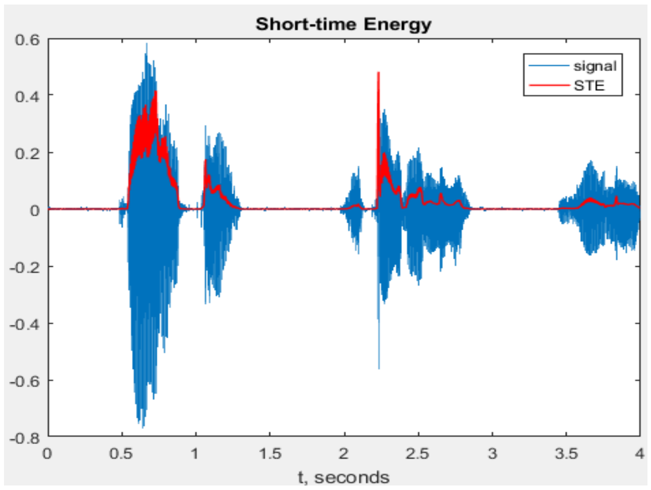


**Figure 10. Plot of the sound energy for a sound wave**

One can notice that energy is high for the “speech” frames and low for the “silence” ones (because in the Energy formula, highest amplitude values contribute mostly, see Figure 10). It follows that one can define a threshold based on which the frames are classified.

However, it is not enough to calculate the energy for removing the silence. In our model for speech recognition we also calculated the *zero crossing rate* (ZCR) for each frame.

$$Z = \sum_{n=1}^{N-1} \frac{1 - sign(x(n))sign(x(n+1))}{2}$$

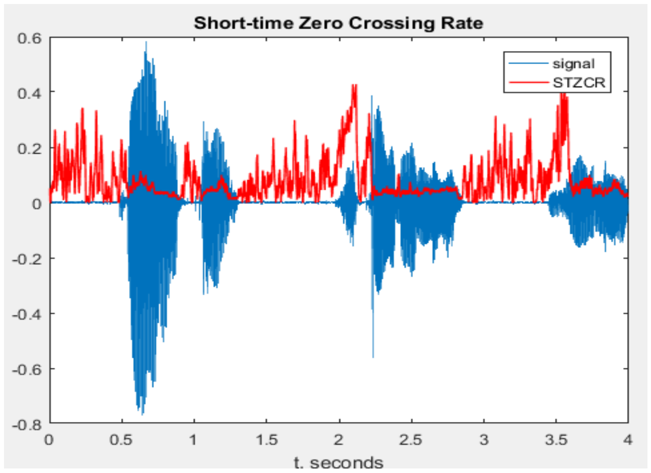


**Figure 11. Plot of the zero crossing rate**

Statistically it was determined that ZCR has higher values in the regions corresponding to silence and a smaller values in the regions corresponding to speech (see Figure 11).

By computing for each frame the ZCR and Energy we were able to remove the silence from each frame[20].

## Fast Fourier Transform

Fourier Transform gives an alternative representation of a function by showing how it can be written as a sum of sinusoidal functions.

*Fast Fourier Transform* is an algorithm used to extract the frequencies from a windowed signal. By running the FFT algorithm we obtained the energy at each discrete frequency band (see Figure 12).

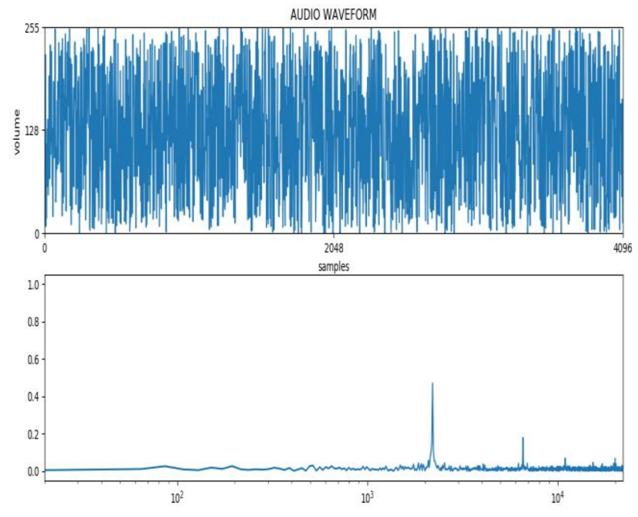


**Figure 12. Time domain vs frequency domain**

The algorithm works with the Complex numbers, because $\boldsymbol{R^2} \approx$ **C.** We assume that each point from $\boldsymbol{R^2}$ has the coordinates ($x$ ,$y$), and each point can be viewed as a complex number $x + i * y$.

The general formula is:

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-\frac{2\pi ikn}{N}}$$

Our speech is a sum of the sinusoids functions. Using the Fourier transform we want to separate them in functions with the same frequency.

Using the Euler's formula $e^{ix} = cosx + isinx$ our formula becomes:

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-\frac{2\pi ikn}{N}} = \sum_{n=0}^{N-1}\left[\cos\left(\frac{2\pi kn}{N}\right) - i * \sin\left(\frac{2\pi kn}{N}\right)\right]$$

Firstly, we create an array of complex numbers where the real part of each complex number is the point from time domain speech, and imaginary part is zero. The algorithm gets as input a vector of complex numbers and returns another vector of complex numbers. We make this because we need to extract the power spectral density from each frame.

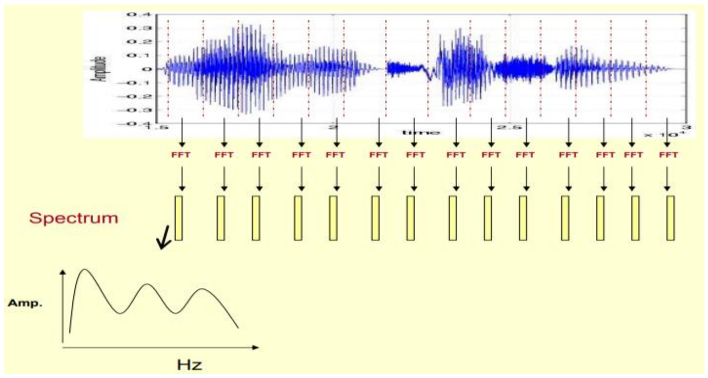


**Figure 13. Power spectral density**

We know that the form of the complex number is $Z=x + i * y$.

To extract the spectrum we just calculate the module of the complex number $|Z| = \sqrt{x^2 + y^2}$ so, we obtain the vector of frequencies for each frame.

## MFCC

An approximation which captures the properties of human hearing perception is represented by Mel scale filtering. To pass from frequency (Hz) to Mell one needs the following formulas:

$$\boxed{F = 700 * 10^{\frac{M}{2595}} - 1} \qquad \boxed{M = 2595 * \ln\left(1 + \frac{F}{700}\right)}$$

The frequencies range resulted from the FFT algorithm is very wide so one has to normalize the voice signal. By applying the Mell-Scale filter bank one converts the frequency to Mell scale[22].

We use these filters to mimic the human hearing.

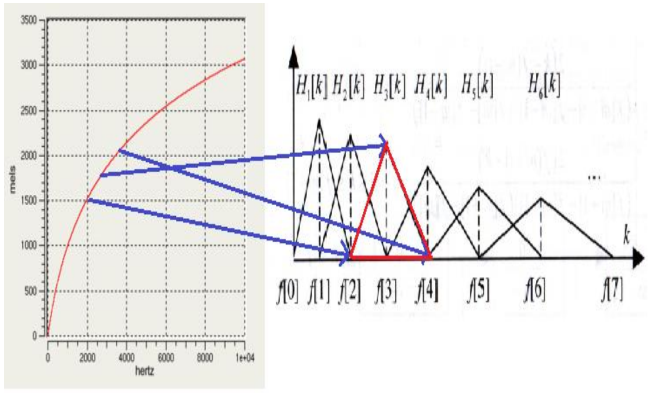

mels
hertz
$H_1[k]$ $H_2[k]$ $H_3[k]$ $H_4[k]$ $H_5[k]$ $H_6[k]$
k
f[0] f[1] f[2] f[3] f[4] f[5] f[6] f[7]

**Figure 14. Mell Filter bank**

We used a set of triangular filters to compute the spectral components. We used 26 triangular filters to get the approximation of the Mell scale. The input for each triangular filter is represented by the corresponding power spectrum. The output obtained after applying a filter is a vector of spectral energies.

After filtering the frequencies one has to sum the coefficients and take the logarithm of their values (this procedure is also motivated by the human hearing because hearing is not linear, by applying the logarithmic function one reduces the amplitudes of the frequencies).

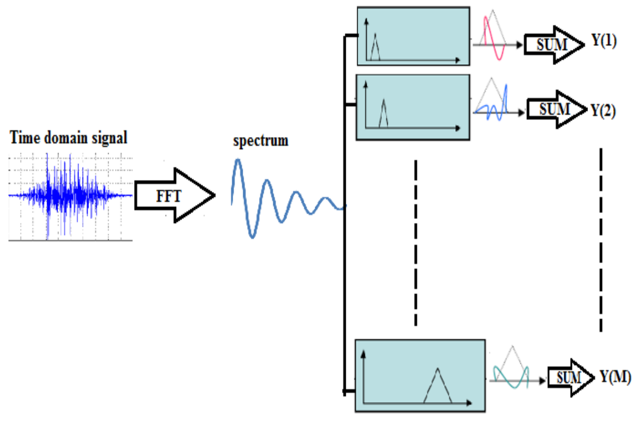


Figure 15. Sum & log energy

## Discrete cosine transform (DCT)

Next step one has to go back from frequency domain to time domain by performing DCT.

$$Y(k) = \sum_{l=0}^{L} \cos[\frac{\pi}{L}(l + \frac{1}{2})m]E_k; \quad 0 \leq m \leq L - 1$$

We compact the energy from Mell scale and we get 26 coefficients. We keep only 12 coefficients from 26, because higher coefficients are characterized by small changes in energy. Dropping them, we get a small improvement.

We get feature vector, size = [number of frames, 12]

This feature vector can be used for pattern recognition, using Dynamic Time Warping or for training a Convolutional Neuronal Network.

## Dynamic Time Warping (DTW)

For recognizing a test file one has to compare the test feature vector with each feature vector from the dataset. However, the test and the training files have different lengths. Consequently, the feature vector is a two dimensional array (number of frames times number of feature points in each frame, i.e. 12). If the vectors were of the same length, one could use Euclidean or cosine norms to determine the distance between them. The problem is how to compute the norm of two time series, which have the different length. Dynamic time warping is a dynamic programming algorithm for measuring the similarities between two time series with different sizes. The algorithm assumes that in each point one has to compute the distance between the test and reference files in the following way:

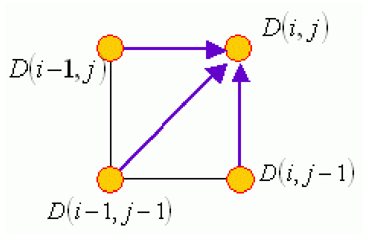


Figure 16. DTW

$$D(i,j) = |t(i) - r(j)| + min \begin{Bmatrix} D(i-1,j) \\ D(i-1,j-1) \\ D(i,j-1) \end{Bmatrix}$$

where t is the test vector, r is the reference vector, and D is a matrix of distances between two time series.

The purpose is to minimize the final D(I,J) which represents the similarities between the test and reference vectors[16].

| Train-Test | Time(minutes) | Accuracy (%) |
|---|---|---|
| 90% - 10% | 4.15 | 100% |
| 80% - 20% | 8.3 | 99.9% |
| 70% - 30% | 12.45 | 99.8% |

Table 1. Testing DTW .

We have noticed that as the number of test files increases, the accuracy rate decreases. This is because the words variability decreases there are some words that are similar to each other.

## SPEAKER RECOGNITION

We shall teach the computer to recognize not just the speech but also the speaker. A half of work is done by feature extraction used by speech recognition. Firstly, we shall compute the FFT for each user. Each user shall speak the same phrase 3 times (we need this to extract user band of frequency)[10]. Then we need to take the maxim amplitude of each frequency, and save the point of frequency where amplitude is maxim.

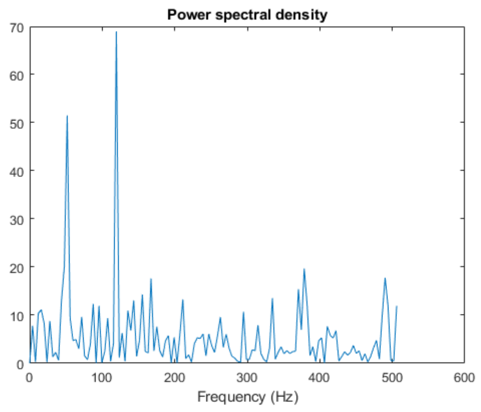


Figure 17. Spectrum of speaker

From the Figure 17 we see the maximum amplitude of frequency. We shall keep the frequency from this point, and use it as feature point for speaker recognition. To recognize someone, we need to compute the distances between test point and each point from dataset, and take the nearest one. To optimize the calculations we can use the K-means clustering 1D. For each user we will have a centroid, which represents all feature points of a specific user. Speaker recognition is done with Nearest Neighbor.

**Artificial Neuronal Network**

The main drawback of the DTW is that it takes a lot of time to compute the similarity between the test file and whole dataset. For example, if the dataset contains 1000 words, it will compute 1000 times. To avoid this, we trained a Convolutional Neural Network model, so that the test file is recognized "instantly".

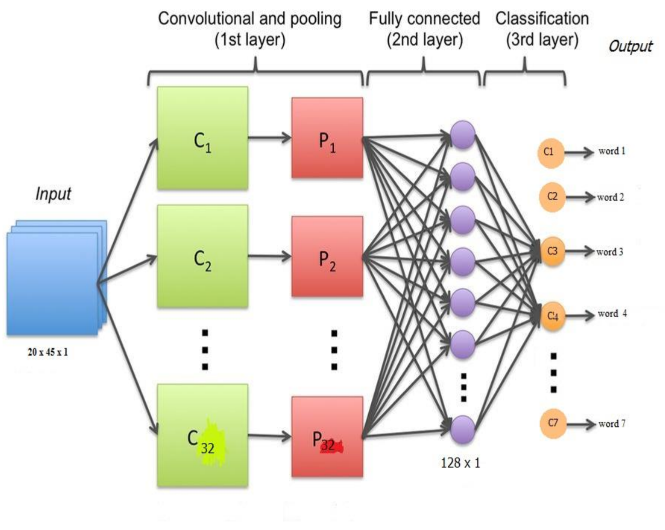


Figure 18. Convolutional Neural Network

We trained the computer to recognize the speech by using a neural net defined as in Figure 18. Our dataset consisted of 700 words (in the future we shall increase it), each word being spoken 20 times by different persons, so we got different pronunciation for each word. For each file from dataset we computed MFCC and extracted feature vectors. Feature vectors represented the input data for our neurons. For Convolutional Network we used TensorFlow with Keras[6]. Firstly, we needed to represent the data in order to train our model. We used one hot encoding from Keras to represent out labels (labels represent words name). We get the unit matrix, with 1 on the main diagonal and 0 in rest. Then we split our data in 80% for training and 20% for testing. First layer is convolutional 2D [11], with kernel size = (2,2) and activation function Rectified Linear Unit (ReLU) [3]. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. Input shape is (20 x 45 x 1), where 20 is the number of feature points from each frame, 45 is the number of frames, and 1 is the number of channels. Each frame has 20 feature points, but not all speech files have the same number of frames. To avoid this inequality, we created the same number of frames for all files, and the free space was filled with zeros. We used ReLU because this activation function makes zero the negative weights while leaving the positive weights as they are (see Figure 19).

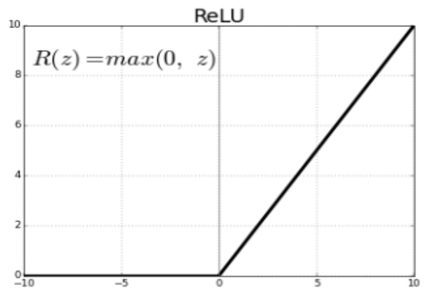


Figure 19. Activation function Rectified Linear Unit

The next layer in our model is MaxPooling2d with pool size = (2,2). It is used to make the down sampling. To avoid overfitting in Keras, we used Dropout[1] with α=20, that is, our model will drop out 20% of weights and will learn only from 80%. Then we used Flatten layer and added a fully connected layer with 128 neurons, the activation function being the same (ReLU). After another Dropout we used a fully connected layer with 7 neurons, (7 because our dataset is composed from 7 words, each spoken by 100 times), activation function being 'softmax' [17].

The Keras model from Figure 17 can be created in Python this way:

```
model =Sequential()
model.add(Conv2D(32,kernel_size=(2, 2),
activation='relu', input_shape=(20, 45, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(7, activation='softmax'))
```

We trained our model with four optimizers and the results are depicted in Table 1.

| Optimizer | Loss (%) | Accuracy (%) |
|---|---|---|
| SGD | 85.13 | 14.86 |
| Adamax | 4,47 | 96.43 |
| Adam | 4.99 | 95 |
| Adadelta | 2.21 | 97.78 |

**Table 2. Training CNN on 150 epochs with different optimizers.**

The loss function is categorical cross entropy, it is objective function for minimize [2]. It is the sum in the smallest squares. It finds network weights to minimize the training error between target and output labels of training examples.

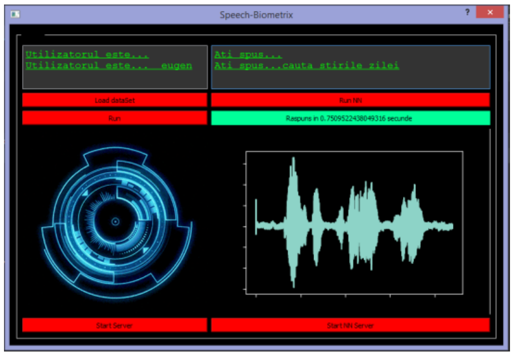
$$E(w) = \sum_{i=1}^{N} (target_i - output_i)^2$$

Next we update weights by gradient descend:

$$w = w - \alpha \frac{\partial E}{\partial w}$$

Where $w$ is a weight, $\alpha$ is learning rate and $E$ is total error.

As optimizer we used Adadelta - an adaptive learning rate method for gradient-based optimization algorithm. We trained our model for 150 epochs, accuracy is 97.78%. An epoch consists of feedforward and backpropagation functions. The training was done locally on the computer's CPU. The application was implemented in Python, and extended on android phone with Android Studio platform, using TCP/IP communication.



Utilizatorul este...
Utilizatorul este... eugen

Ati spus...
Ati spus...cauta stirile zilei

Raspuns in 0.7509522438049316 secunde

**Figure 20. Graphic user interface of the application**

To compile our model in Keras:

*model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])*
*model.fit(X_train, y_train_hot, batch_size=30, epochs=150, verbose=2, validation_data=(X_test, y_test_hot))*

The application allows the user to interact with the computer by using voice authentication and several voice commands.

## CONCLUSION

We observed that the system was very fast when the Convolutional Neural Network was used (less than 1 second for recognizing a phrase on a mid-range personal computer). However, CNN accuracy is not 100%. Using DTW we achieved 100% recognition, but it takes a lot of time. DTW have to match test file with each file from dataset, and if our dataset contains 1 million patterns, the algorithm will check all of them. To improve our score, we need to increase our dataset, more people to record more data. Another solution is to combine DTW with Neural Network. The Convolutional Neural Network must learn how to compare two time series and get the best score in less time. In Table 2 we compared the execution time and the recognition rate for DTW and CNN. The testing files are randomly selected from the dataset. We have noticed that by increasing the number of test files, the recognition rate decreases for both DTW and CNN.

| Number of samples | Time (minutes) | | Accuracy (%) | |
|---|---|---|---|---|
| | DTW | CNN | DTW | CNN |
| 50 | 3.05 | 0.25 | 100 | 98.5 |
| 100 | 6 | 0.51 | 100 | 97.8 |
| 200 | 13 | 1 | 99.95 | 97.8 |
| 500 | 30 | 2.5 | 99.9 | 97.3 |

**Table 2. Time and Accuracy DTW vs CNN**

The final conclusion is that DTW is good to use for small data set, as long as the recognition rate is about 99.9%, the runtime for the small set of data is acceptable. When we have larger datasets, it is recommended to use a convolutional neural network because its recognition rate can be improved as the number of training files increases.

### Feature work

In the future, we plan to increase the number of words recorded in the dataset. We will model a neural network for recognizing the speaker and add more speakers to the data set.

## ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0917 / contract no. 21PCCDI/2018, within PNCDI III.

## REFERENCES

1. Lee, C.-y., Gallagher, P. W. *Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree.* In AISTATS, (2016).
2. Ioffe, S. and Szegedy C. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. In ICML, (2015).
3. Xu, B., Wang, N., Chen, T. *Empirical evaluation of rectified activations in convolutional network*. In ICML Workshop, (2015).
4. Campbell W., Gleason T., Advanced Language Recognition using Cepstral and Phonetic: MITLL System Performance on the NIST 2005 Language Recognition Evaluation, *Speaker and Language Recognition Workshop, IEEE Odyssey*, (2006),1-8
5. Espy-Wilson C., Manocha S., A new set of features for text independent speaker identification, *INTERSPEECH - ICSLP*, (2006), 1475–1478.

6. Keras: The Python Deep Learning library. https://keras.io/#keras-the-python-deep-learning-library

7. Furui S., *Digital Speech Processing, Synthesis and Recognition*, CRC Press, USA, (2000).

8. Gajic B., Paliwal K., Robust parameters for speech recognition based on subband spectral centroid histograms, *Proc. 7th Eur. Conf. Speech Commun. Technol. EUROSPEECH*, Aalborg, Denmark, (2001).

9. Glass J., Victor Zue, *Speech Recognition*, Open Course Ware, MIT, (2003).

10. Kinnunen T., *Spectral Features for Automatic Text-Independent Speaker Recognition*, University of Joensuu, Dpt. of Computer Science, Finland, (2003).

11. Knill K., *(Deep) Neural Networks for Speech Processing*, Cambridge University Engineering Department, (2015).

12. Linde Y., Buzo A., Gray R.M., An Algorithm for Vector Quantizer Design, *IEEE Transactions on Communications*, 28, (1980), 84–95.

13. Popovci D.M., Zaharescu E., Rusu A., Puchianu C.M., Sburlan D., *Medii virtuale multimodale distribuite*, vol. III., Editura Universitaria, Craiova, (2015).

14. Rabiner L.R., Juang B. H., *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, N.J., (1993).

15. Rabiner L. R., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proc. of the IEEE*, 77, 2 (1989), 257–286.

16. Salvador S., Chan P., Toward Accurate Dynamic Time Warping in Linear Time and Space, *Journal of Intelligent Data Analysis*, 11, 5 (2007), 561–580.

17. Soltau H., Saon G., Sainath T.N., Joint Training of Convolutional and NonConvolutional Neural Networks, *Acoustics, Speech and Signal Processing (ICASSP)*, (2014), 5609–5613.

18. Saheli A. A., Abdali G.A., Abolfazl A., Speech recognition from PSD using neural network, *Proc. of the International MultiConference of Engineers and Computer Scientists*, vol. 1, (2009), 174–176.

19. Tiwari G., Pandey M., Shrestha M., *Text-Prompted Remote speaker authentication*, Department Of Electronics And Computer Engineering, Nepal, (2011).

20. Short-time Energy and Zero Crossing Rate. https://www.mathworks.com/matlabcentral/fileexchange/23571-short-time-energy-and-zero-crossing-rate?focused=3805785&tab=function visited on 05.05.2018

21. Brigham E., *Fast Fourier Transform and Its Applications*. Prentice-Hall, USA, (1988).

22. Black W.A. Speech Processing Lecture notes Fall, Carnegie Mellon, (2011).

23. Burileanu C., Popescu V., Buzo A., Petrea C.S., Ghelmez-Hanes D. *Spontaneous Speech recognition for Romanian in spoken dialogue systems*. PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A, Volume 11, Number, (2010), 83–91.

24. Denilson C.S., *A Robust Endpoint Detection Algorithm Based on Identification of the Noise Nature*, ITRW on Nonlinear Speech Processing (NOLISP 07) Paris, France May (2007), 22-25.