AGH University of Science and Technology

Voice command recognition system

Krzysztof Belcarz, Maciej Cebula, Marcin Kowalczyk, Kamil Lelowicz, Piotr Merynda, Damian Paciuch, Maciej Podsiadło, Łukasz Radzio, Karolina Szmyd, Ignacy Solecki

Kraków, 2016

Abstract

This paper presents work on a simple voice command recognition system, which was designed and implemented as part of a class science project.

Taking under consideration how complex the speech signal is, the solution proposed in the paper allows for recognition only of short, direct commands that could be used, for example, to steer a motor device. It is based on the LPC (Linear Predictive Coding) for cepstral coefficient extraction and DTW (Dynamic Time Warping) algorithm for signal classification. System is implemented using Python.

1. Introduction

As the progress of technology and science continues, voice communication between man and machine becomes more and more desired. It has already been introduced to public use in some of the most basic, daily activities, i.e. using one's phone. However, scientists still struggle to ensure the highest quality of speech recognition, along with its reliability. One of the applications of presented feature that comes easily to mind, would be a simple, voice controlled video game, as the project recognizes four basic commands: left, right, start and stop. Another possible use, if some development is instated, is connected to a recently trending research subject of autonomous cars. It is obvious that dependability of voice recognition would be the highest priority in such projects, as any mistakes could potentially lead to a disaster.

2. Algorithmic description

Overall process of signal computation is presented on Fig. 1.

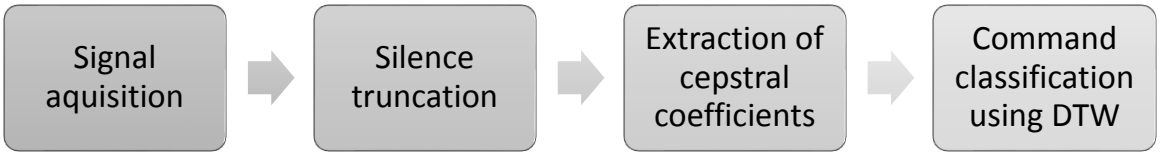

Figure 1: Signal processing scheme

First step in voice recognition is the analysis and initial processing of the signal. In the proposed system, it constitutes of eliminating any unnecessary silence before and after the actual command. In order to achieve that, the signal is normalized and then its energy is computed, in 10ms frames. Afterwards, n frames from the beginning and z frames from the end are removed, if their calculated energy is less than an appointed threshold value – $0.25 * fr/8000$ (fr – signal sampling frequency, 44kHz). Fig. 2 presents how this step improves the input signal.

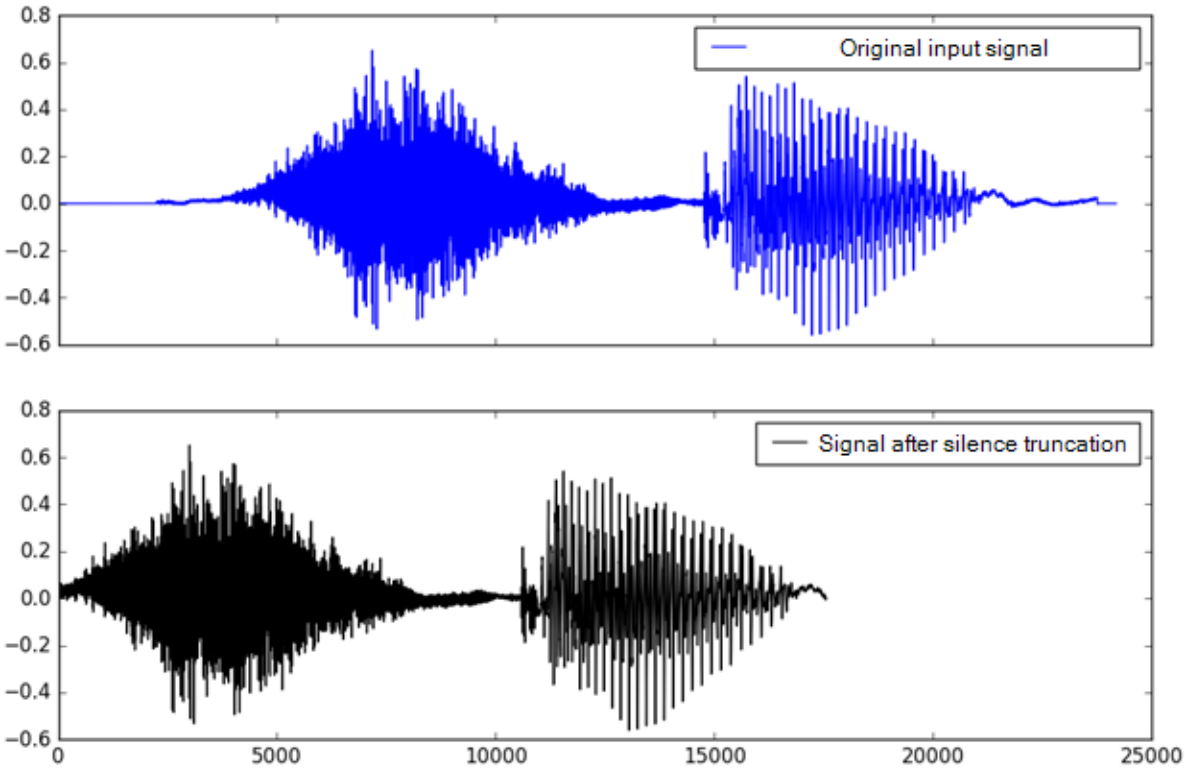

Figure 2: Presentation of silence truncation method

The next operation that the signal is subjected to, is a pre-emphasis. This procedure ensures that the signal-to-noise ratio is improved, by means of increasing quantization components at high frequencies of the analog audio signal, using the derivative of input signal. Subsequently, signal is divided into 30 ms time frames in 10 ms increments. Then, mean frame value is removed and Hamming window function is applied. After these operations, each frame is now properly pre-processed and ready for coefficient extraction. Linear description coding (LPC), which is used in the project, allows to describe each frame with a 12-element coefficient matrix, that in later steps will be used for comparison between input signal and the signal database. Number of

extracted coefficients can be modified, but the minimum suggested value is 8. Last step of the algorithm – command recognition by comparing its coefficient matrix to the model database – uses the DTW (Dynamic Time Warping) method. It is in general a class of algorithms used for time series comparison. It allows to find the smallest distance between two series, with permission to transform times for both series. If two signals are given x of length M, and y of length N, DTW calculates a matrix of size $N \times M$, where each of the elements amounts to the distance between series elements (respectively $n = 1,2,\dots,N$ and $m = 1,2,\dots,M$). Thereafter, a path between points $(1,1)$ and (N,M) is computed, such that its cost is the lowest possible. Here, the longest common subsequence (LCS) method was implemented. Its result is the cost of the shortest path for each model and the index of the smallest matrix element.

3. Implementation and tests

The project was implemented using Python version 3.5.2 and the following libraries:

- math – use of math functions, e.g. sin(), max(), min();
- numpy – allows for creation and operations on vectors and matrices;
- scipy.io – allows for the input of data files in the form of vectors.

Benchmark database consisted of voice command samples, prepared by one person. There were multiple inputs for each of four commands, registered in Polish ("lewo" – left, "prawo" – right, "start" – start and "stop" – stop).
Each registered input signal undergoes the process described in the previous section. The result of it is a matrix of distance between the input signal and the four base samples. The smallest distance and its index is the result of classification.

Tab. 1 presents the results of conducted tests. Each row should be understood as follows: [input sample] is: [recognized sample] [matrix od distance]. As we can see from the results, for provided benchmark database, the algorithm had an efficiency of 100%. What should not be ignored, however, is the fact that this result could greatly vary in regards to the sample quality, both benchmark and input ones.

left is: left	[0.	2.29052189	4.13228302	5.81446397]
left is: left	[2.40757516	2.51416631	4.33090219	5.63726275]
left is: left	[2.09952197	2.71061027	4.05510047	5.65013557]
left is: left	[2.3842971	2.6831393	4.14077782	6.14104141]
right is: right	[2.57421985	0.	4.56611576	7.42960373]
right is: right	[2.78624717	2.30245791	4.20277071	6.95717671]
right is: right	[2.73850642	2.18775838	4.69807601	7.67753523]
right is: right	[2.44346479	2.12889944	4.54110114	7.68231522]
start is: start	[3.86584299	3.29568316	0.	4.36473288]
start is: start	[4.05555847	3.43467927	2.52675626	4.19724796]
start is: start	[4.02033715	3.74007073	2.8279758	4.24157431]
start is: start	[4.30684637	3.67073155	3.31277276	3.61667766]
stop is: stop	[6.08212694	7.03294449	4.72430429	0.]
stop is: stop	[4.27812437	4.40001481	3.55378665	3.24485962]
stop is: stop	[4.8773367	4.38359242	3.89180511	2.46823679]
stop is: stop	[4.4823141	4.4754866	3.66997733	3.27118922]

Table 1: Test results

4. Conclusions

The paper has presented a successfully executed project of basic voice command recognition system. Properly implemented algorithm can now be a base for further development, in order to introduce more commands to be recognized. One of the possible improvements would be the reduction of noise and influence of the general environment, in which the samples are being registered, on the algorithm’s results.