Recognition of Voice Commands Based on Neural Network

Vyacheslav Lyashenko [1], Farah Laariedh [2], Svitlana Sotnik [3], M. Ayaz Ahmad [2]

[1]*Department of Informatics, Kharkiv National University of RadioElectronics, Ukraine*
[2]*Department of Physics, Faculty of Science, University of Tabuk, Saudi Arabia*
[3]*Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of RadioElectronics, Ukraine*

Abstract **– The paper considers features of voice commands pronunciation models, namely, dependence: system accuracy on number of states for phonemes; system accuracy on learning rate; accuracy of system on value of training set. A speech recognition system based on neural networks is proposed. A speech recognition system is not easy to implement and requires an understanding of speech recognition basics. The developed system is compared with Speech Recognition from Google and Pocket Sphinx. The proposed system can recognize voice commands with an accuracy of 84.4 %.**

Keywords **– intelligent, system, recognition, voice, commands.**

1. Introduction

Currently existing systems with voice command recognition (VCR) open up new possibilities and are widely used by completely different users, starting from medicine, for example, doctor in polyclinic can pronounce diagnoses that will immediately be entered into an electronic card and ending with Internet of things systems (IoT).

DOI: 10.18421/TEM102-13
https://doi.org/10.18421/TEM102-13

Corresponding author: M. Ayaz Ahmad,
Department of Physics, Faculty of Science, University of Tabuk, Saudi Arabia
Email: a.ahmad@ut.edu.sa

Received: 03 February 2021.
Revised: 02 May 2021.
Accepted: 10 May 2021.
Published: 27 May 2021.

Such systems can provide:

- safe work due to ability of control systems using voice commands;
- voice text input;
- voice search, etc.

For example, IoT systems offer large number of interaction options, one of which is voice command recognition.

Advances in speech recognition and synthesis have fueled emergence of voice assistants that can communicate with user and execute various voice commands. This communication option increases productivity in production, due to convenience of interacting with intelligent systems [1].

A neural network can be used as tool for implementing voice command recognition systems. Neural networks can perform various tasks, including voice recognition, and have some advantages over traditional methods (hidden Markov models, sliding window, and placeholder models). With help of neural networks, adaptive systems can be implemented, with constant learning opportunities. Due to this, neural networks are gaining high popularity, since they can "learn" to solve any problem.

The use of voice recognition system in production will increase productivity and improve human-system interaction.

2. Materials and Methods

2.1. Related Work

By now, significant number of works have appeared on recognition of voice commands, which deal with methods of teaching voice control [2], machine learning algorithms for converting text into voice [3], [4], methods of voice conversion [5], [6].

The works [7], [8] describe existing speech recognition systems: Microsoft API, Google API and CMU Sphinx.

In [7], emphasis is on open source speech recognition in different environments – these are audio recordings selected from different sources, calculating the error rate in words.

In [8], it refers to both open source systems (CMU Sphinx, Kaldi, Julius, HTK, iAtros, RWTH ASR and Simon) and closed source systems (Dragon, Mobile SDK, Google Speech Recognition API, Siri, Yandex SpeechKit and Microsoft Speech API).

The authors have widely disclosed the area of application for voice recognition systems to control smart home [9]. The paper describes features of controlling lighting and electrical appliances in home or office using voice commands. Automation is based on voice command recognition and uses low-power RF wireless modules and microcontrollers.

Speech recognition and voice separation for Internet of Things is described in [10], where a low-cost and customizable IoT prototype with support for voice (speech-to-text Google Cloud) was implemented on basis of Raspberry Pi motherboard. In [10] proposes method for solving problem of voice split in context of IoT, when two people try to send commands to voice-enabled IoT device, and both commands have to be processed and executed simultaneously.

In [11], information on use of voice recognition systems for controlling robot is presented. The connection between android app and vehicle is facilitated with Bluetooth technology. The commands from app are converted into digital signals by Bluetooth RF transmitter for corresponding range (about 100 meters) to robot.

The implementation of control using voice commands in modern production was considered by authors in [12], [13], namely, Pick-by-Voice technology, which provides direct voice communication with warehouse management systems (WMS), that is, information transfer without formalizing it in electronic or paper form, and coordinated execution of important warehouse operations number.

In [14], possibility of using voice control of an industrial robot in production using gestures and voice commands in offline and online mode is discussed. The elements of creating software for controlling robot in offline and online modes are described. The application for Kinect module was developed in C # in Visual Studio environment, and industrial robot control program was developed in RAPID language in RobotStudio environment. The applied solution allows robot to work in industrial conditions without negative impact of communication task on time of robot's work cycles.

Human interaction with autonomous robotic assistants is presented in [15], where voice commands are one of most important natural ways of human-robot dialogue.

2.2. Analysis of fFeatures of Voice Commands Pronunciation Models

The performance of VCR can be improved by improving specifics of its speech models.

Speech model comprises implementation of person's dialogue in specific situation of speech communication.

The speech model is whole family of sounds, sometimes very different in composition of vector-signs.

Biphones (which describe phoneme in combination with previous or subsequent phoneme) are used to describe beginning or end of speech fragment, as well as when there was not enough data to construct states of triphon. The contextual phonemes that have been discussed so far are referred to by analogy as monophones.

There are following ways to increase specificity of speech patterns:

- Increasing number of phonemes, for example, by splitting, phonemes and independent processing in dictionary;
- Increasing number of states in background, for example, from 1 to 3 states for one phoneme;
- Dependence on context of phonemes, for example, use of triphones (all combinations of phoneme under consideration with preceding and following sounds as separate acoustic objects for which you need to build your states);
- Modeling of words pronunciation variations, for example, inclusion of several variants of pronunciation in dictionary.

Optimization of degree of speech models specificity of certain database (DB) is time-consuming process and this does not apply specifically to neural networks. Phoneme topology during training is usually negligible, but sometimes up to 3 states per phoneme are used with simple transition from left to right.

During training, about 6300 training sentences from database and 390 can be applied to check accuracy of system from 1 to 3 states per phoneme.

During training, about 6300 training sentences from database and 390 can be applied to check accuracy of system from 1 to 3 states per phoneme.

For English language, number of phonemes is about 50 (number is not fixed – number of common biphons or triphons can be attributed in advance to separate phonemes).

Table 1 shows an example of system accuracy dependence on number of states per phoneme [15]. This Table shows that only 5 epochs were taken, since one epoch leads to underfitting, and an excess of epochs leads to overfitting.

With an increase in number of epochs, weights of neural network change more and more times.

Table 1. System accuracy dependence on number of states for phonemes

Epochs	Word accuracy, %		
	1 state per phoneme	1…3 states per phoneme	3 states per phoneme
1	81.9	85.1	85.0
2	85.8	86.4	86.2
3	86.3	86.9	87.0
4	86.3	87.0	88.0
5	86.1	86.9	88.0

Based on Table 1, we can say that with increasing iterations, greater word accuracy was obtained with 3 states per phoneme (88.5%), and lowest accuracy with 1 state per phoneme.

After analyzing Table 1 for 1 ... 3 states per phoneme, it can be seen that initially accuracy is higher and at last iterations accuracy still remains slightly higher than with 1 state per phoneme.

That is, best results were obtained when additional conditions per phoneme were useful and were appropriately trained.

It can be said unambiguously that as states increase, accuracy of dictionary words increases. Flexibility of vocabulary also affects improvement of results of robots VCR, where each word has several pronunciation options. This makes it possible to ensure greater accuracy in recognition of VCR phonemes. Using this technique, you can see that some words have very similar pronunciation therefore, system can often make mistakes on them.

Backpropagation algorithm was used to train network.

Learning rate is critical parameter when learning a neural network.

If speed is not high, network will "learn" very slowly, especially since there is also a direct dependence on size of dictionary.

However, if learning rate is too high, then system may face problem of drop in accuracy at output, since it will not be able to process data well.

Many factors can influence the optimal learning rate, therefor learning rate is selected experimentally (Table 2) [15].

In Table 2 there is already larger than number of epochs – 10. Often number of epochs is associated with diversity in data.

For example, Table 2 shows dates in terms of frame precision and word precision.

The word accuracy dates show that at lowest speed 0,0003 is too small and the accuracy (10 %) is at first iteration, but with increasing iterations, accuracy increases dramatically and still remains unstable closer to 8th iteration, however, frame accuracy dates although it is in range of 55-60 %, it is relatively stable.

Table 2. System accuracy dependence on learning rate

Epochs	Accuracy, %							
	Word accuracy				Frame accuracy			
	learnRate .0003	learnRate .001	learnRate .003	learnRate .01	learnRate .0003	learnRate .001	learnRate .003	learnRate .01
1	10	70	81	76	55	60	60	55
2	44	78	86	78	57	61	61	55
3	68	83	84	75	59	60	60	56
4	78	84	84	72	60	61	60	54
5	80	84	84	74	60	60	60	55
6	81	84	84	70	59	60	59	55
7	81	87	86	73	60	60	59	54
8	83	84	84	75	60	60	59	56
9	80	86	85	72	60	60	59	54
10	81	87	86	75	60	60	59	56

If we consider dates of accuracy at highest learning rate of 0,01, then we see that accuracy of frame (55-56 %) and words (72-75 %) is not very high and optimal (network fluctuates).

From Table 2 it can be seen that word accuracy improves at learning rate of 0,001 already at last epochs, which cannot be said for frame accuracy, although it also slightly improves compared to learning rates of 0,01, 0,003.

Also from Table 2 it can be seen that at learning rate of 0,003, accuracy is initially optimal and then also slightly increases.

If two different networks are trained at same learning rate, it would be unfair to compare their results after fixed number of iterations, because learning rate dates could be optimal for one of networks, but not optimal for other [15].

That is, learning rate should decrease over time so as not to disturb network too much as it approaches optimal solution.

If it is necessary to reduce learning rate, one should take basic learning rate, if it is high, and geometrically reduce it using coefficient that is less than 1 after each training epochs.

Dates show examples of adjusting learning rate using coefficient that ranges from 0.5 to 1.

Thus, coefficient that is equal to 1, on contrary, leads to too high level of learning, so system becomes unstable.

The best result is mean, which gives enough time for system to break out of local minima even before learning rate efficiency drops to 0 [15].

Factor of 0.5 gives best training accuracy, but this advantage is soon lost, since learning rate decreases rapidly, and system cannot get out of local minima (Table 2).

Although geometric coefficients are important in determining learning rate, this is not main point, since it is impossible to know for sure that system has reached its peak accuracy on these inputs.

The following factors influence optimal learning rate to lesser extent:

1. Size of training set, namely, larger training set provides for lower learning rates in each iteration (Table 3) [15].

This is because optimal learning rate dates decreases slightly after each change in weighting factor.

Model is set of data. If model is complex, then it is divided into training suites and test suites. Remember that training dataset is data on which model will learn, and test dataset is data on which model will be tested after training.

Amount of training data should always be greater than test data. They usually occupy 60-80 % of dataset.

Table 3. Accuracy dependence of system on size of training set

Epochs	Optimal learning rate					
	100 training sentences	200 training sentences	500 training sentences	1000 training sentences	2000 training sentences	3600 training sentences
1	0.04	0.03	0.02	0.01	0.01	0.01
2	0.011	0.0075	0.006	0.006	0.006	0.0025
3	0.005	0.0035	0.003	0.002	0.002	0.0025
4	0.0025	0.002	0.001	0.001	0.001	0.0015
5	0.002	0.0025	0.001	0.001	0.001	0.002
6	0.001	0.0015	0.001	0.001	0.001	0.0015

2. Normalization of inputs, namely, higher standard deviation of input, allows high learning rates to compensate for fact that hidden neurons are saturated.

3. Activation function – function that calculates output of an artificial neuron.

4. Number of neurons on input and hidden layers.

In Table 3 shows that with an increase in training set of phrases, optimal learning rate decreases, but regardless of training set size, learning rate decreases during training. But it should be borne in mind that optimal learning rate dates attenuates slightly after each weight update.

Thus, it was determined that dates of optimal learning rate are more influenced by learning algorithms. A large training set often needs to use lower learning rate per epoch, but learning rate does not have to be constant throughout entire training procedure, and high learning rates can lead to network fluctuations.

3. Development of an Intelligent Voice Command Recognition System

Artificial neural networks are based on structure of simple nonlinear computational elements.

RNM (Recurrent Neural Networks) is class of artificial neural networks, connections between nodes of which form graph oriented in time. This creates an internal state of network that allows it to exhibit dynamic behavior over time [16].

In general, deep neural networks and recurrent neural networks are considered as most modern method of automatic speech recognition. An element (also called a node) of this type is shown in Figure 1 [17].

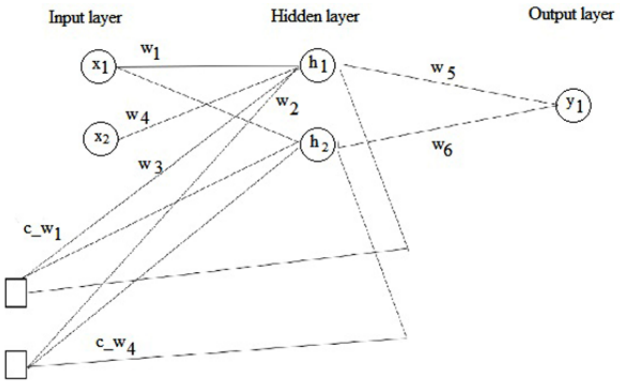

Figure 1. Nodes of neural networks

The node has N inputs, denoted x_1, x_2, …, x_N, which are summed with weights w_1, w_2,…, w_N.

First, information is sent from input layer by weights to hidden one:

$$h_1 = (x_1 \cdot w_1) + (x_2 \cdot w_4),\ h_2 = (x_1 \cdot w_2) + (x_2 \cdot w_3),$$

where:

- h – hidden layer.

Then information is sent from hidden neurons to time delay layer and to network output:

$$c_1 = h_1, c_2 = h_2;$$

$$y_1 = (h_1 \cdot w_5) + (h_2 \cdot w_6).$$

Next, data is written to time delay layer, and then signal is "run" again, only signals from time delay are added:

$$h_1 = (x_1 \cdot w_1) + (x_2 \cdot w_4) + (c_1 \cdot c\_w_1) + (c_2 \cdot c\_w_3),$$

and on second hidden neuron

$$h_2 = (x_1 \cdot w_2) + (x_2 \cdot w_3) + (c_1 \cdot c\_w_2) + (c_2 \cdot c\_w_4).$$

Then received data is sent again to delay layer and to output:

$$c_1 = h_1, c_2 = h_2;$$
$$y_1 = (h_1 \cdot w_5) + (h_2 \cdot w_6).$$

The neuron has weights that are multiplied by received data, resulting in modified response.

Then modified responses in neuron are added up and go to activation function.

The activation function makes answer out of sum.

You can use threshold function or sigmoidal (hyperbolic tangent and logistic function).

Threshold function include the following: when there is summation result and some threshold, it is necessary to compare them. If total result is greater than threshold, then neuron will output 1 and if not, then 0.

The hyperbolic tangent converts the total result to number between -1 and 1. To do this, use formula [17]:

$$\exp(out) - \exp(-out) / \exp(out) + \exp(-out),$$

where:

- exp – exponential function.

The logistics function converts total result to number from 0 to 1. To do this, use formula [17]:

$$1/(1 + \exp(-out)).$$

Ultimately, it turns out that recurrent neural networks are capable of short-term memory. Many tools can be used to solve speech classification problem.

For implementation of software, Python was chosen as programming language.

Python is high-level object-oriented programming language with strong dynamic typing [18], [19].

This programming language has following advantages:

- convenient tool for solving math problems;
- syntax;
- large number of third-party libraries;
- open source.

To solve speech classification problem at first stage, you should choose library with training data. In order for network to have high accuracy, sufficiently large training database is needed.

TIMIT library was used to solve speech classification problem. This library was developed to provide speech data for acquisition of acoustic-phonetic knowledge and for development and evaluation of automatic speech recognition systems.

TIMIT is library that was developed for research projects in defense and information science and technology management [20].

Phonemes digitized at 20 kHz with 10 kHz tuning filter. After that, phonemes were filtered out and reduced to 16 kHz.

Each of phonemes has label. These labels represent somewhat intermediate level of information between phonemic and acoustic.

RNNLM framework (recurrent neural network language models) was used to train network.

Unlike feed forward neural networks, RNMs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as non-segmented continuous handwriting recognition and speech recognition (Figure 2).

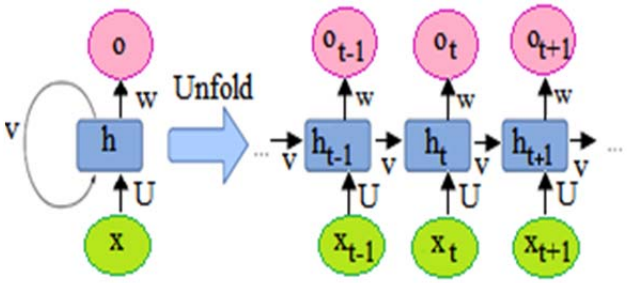

Figure 2. Architecture of recurrent neural network model

As described above, that is recurrent neural network model whose input layer uses 1-of-N representation of previous word that is combined with previous hidden layer state of s(t) using sigmoidal activation function.

Output layer y(t) has same dimension as w(t).

Learning will take place according to stochastic gradient algorithm.

Obtaining weight coefficients in neural network can be modeled as nonlinear global optimization problem.

Objective function for assessing fitness or error of certain weight vector can be formed as follows:

1. Weights in network are set according to weight vector.
2. Network is judged by training sequence. Typically, sum of squared differences between predictions and target values specified in training sequence is used to represent error in current weight vector.
3. To minimize this objective function, one can apply arbitrary methods of global optimization [12].

In Figure 3 shows block diagram of developed program algorithm.

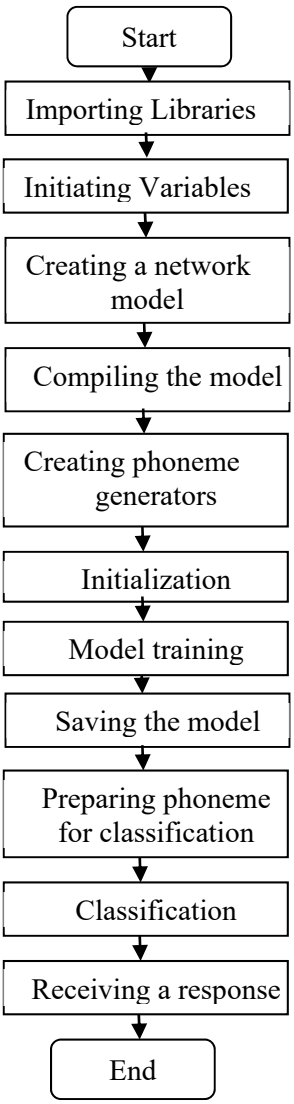

Figure 3. Block diagram of program

Program uses recurrent neural network to solve the problem.

Kaldi library was used to write program.

Program is decomposed into several blocks, each of which is responsible for its own functionality.

Project structure is as follows:

- start_recognition – script for starting recognition procedure;
- / tools – set of tools for recognition;
  data_preparator.ru – script for preparing data for recognition;
  recognizer.ru – speech recognition script;
  segmenter.ru – speech segmentation script;
  transcriptins_parser.ru – script for parsing recognition results;
- / model – set of files for recognition model;
- / web – web application with speech recognition demo stand.

The Alphacep model is used as an acoustic model.

If necessary, you can use your own model. To do this, you just need to replace files in / model root.

After importing libraries, you need to implement function to start speech recognition (Figure 4).

This function initializes variables for working with phonemes in .wav format.

For convenience, logger function was also written, which helps to "identify" errors in program, and if it was not possible to get phoneme to input of neural network (Figure 5).

```
def start_pipeline(wav):
    """
     Launching the speech recognition papline
     Arguments:
         wav: Path to .WAV audio file
    """
    wav = prepare_wav(wav)
    wav_name = Path(wav).name
    wav_stem = Path(wav).stem
    temp = str(Path(TEMP_DIR) / wav_stem)
    os.makedirs(temp, exist_ok=True)
    wav_scp = str(Path(temp) / 'wav.scp')
    make_wav_scp(wav, wav_scp)
```

Figure 4. Function of pronunciation recognition start

```
def terminate_pipeline(is_error, message):
    if is_error:
        LOGGER.error(message)
        os.rename(wav, str(ERROR_DIR / wav_name))
    try:
        delete_folder(temp)
    except:
        LOGGER.error("Failed to delete temporary files for '{}'".format(wav_name))
```

Figure 5. Logger function for working with errors

Entire program is built using error handling using the try… catch syntax.

After program receives files with phonemes, file segmentation starts.

If file is not segmented, corresponding message is displayed in terminal and system tries to receive segments. If it was not possible to receive segments, then message is displayed and then program stops working (Figure 6.).

```
try:
    LOGGER.info("Starting file segmentation'{}'".format(wav_name))
    segm = segmenter.Segmenter(wav_scp, SEGM_MODEL, SEGM_POST, SEGM_CONF, temp)
    segments = segm.segment()
    LOGGER.info("Finishing file segmentation'{}'".format(wav_name))
except:
    terminate_pipeline(True, "Failed to execute file segmentation
    '{}'".format(wav_name))
    return
if os.stat(segments).st_size == 0:
    terminate_pipeline(True, "In file '{}' no segments ".format(wav_name))
    return

try:
    LOGGER.info("Starting to extract segments from a file'{}'".format(wav_name))
    wav_segments_scp, utt2spk, spk2utt = segm.extract_segments(segments)
    LOGGER.info("Finish extracting segments from file'{}'".format(wav_name))
except:
    terminate_pipeline(True, "Failed to extract segments from file
    '{}'".format(wav_name))
    return
```

Figure 6. Block of code for file segmentation

After segmentation, file is recognized using recognizer and transcriptions, and then, subtitles for phoneme files should be generated.

After end of subtitle formation procedure, transcription procedure for phoneme files should be carried out (Figure 7).

```
try:
    LOGGER.info("Start parsing transcription for a file '{}'".format(wav_name))
    pars = transcriptions_parser.TranscriptionsParser(str(OUTPUT_DIR / 'ass'),
    OUTPUT_DIR, LOGGER if IS_LOG else '', 1, 1, CSV)
    pars.process_batch_files([ass])
    LOGGER.info("Finish parsing transcription for a file '{}'".format(wav_name))
except:
    terminate_pipeline(True, "Failed to parse file transcription
    '{}'".format(wav_name))
    return
```

Figure 7. Start phonemic transcription parsing

Also, function of deleting phoneme file after finishing working with it was implemented.

After preparatory actions with phonemes, main block of program starts with start of speech recognition procedure.

Further, paths to phonemes, model file, general graph, text file, recognition configuration file and vector extractor, path to model file, configuration file and posterior segmentation probabilities, etc. were written.

Then arguments are parsed and data is prepared.

4. Results of Experimental Research and Comparison with Analogues

Proposed system has analogues that are implemented according to similar principle and can perform similar tasks.

However, this system is implemented using maximum flexibility of system so that it can be easily adapted to task at hand.

Compared to Google's Speech Recognition system, proposed system does not require permanent connection to Internet.

Autonomous operation of voice command recognition system has advantage of lower requirements for working with it.

However, there is also minus in this implementation – Speech Recognition system, thanks to constant connection to Internet, uses cloud technologies, and this has positive effect on speed of system, ease of adaptation it takes several times less time to reconfigure system, since it does not require ugly learning.

It should also be noted that system from Google has ability to continuously learn by updating database. Having large capacities and cloud technologies, system can learn in real time and easily adapt to new speaker.

Proposed system has only one hidden layer, which is sufficient for solving small range of this system tasks. Increasing number of hidden layers also increases time it takes to train network. At same time, similar system uses power of cloud technologies and system reconfiguration occurs instantly.

Accuracy of developed system reached 84,4 % when training system for 10 iterations.

Speech Recognition system shows result of accuracy up to 98 % [21], [22].

It can be concluded that use of cloud technologies can bring system more accuracy and speed of operation however, it will impose some restrictions on it in areas of use.

In comparison with similar PocketSphinx system, which is also standalone and does not use cloud technologies, which are inherent in Speech Recognition system, proposed system showed best result.

Accuracy of PocketSphinx system reached only 79,2 % with similar parameters during training of systems [23], [24]. This is primarily due to use of various neural network topologies.

It should also be noted that systems of large corporations such as Microsoft. Google and Yandex require licensing, and all the code is not available.

While Kaldi library used allows the system to be used under the Apache 2.0 license, it imposes almost no restrictions on it [25].

Proposed system has shown optimal accuracy results when using Kaldi library.

After experiment and processing of results, following table was obtained (Table 4).

Table 4. Comparison results for accuracy and speed

System	Developed system	Pocket Sphinx	Speech Recognition
Recognition accuracy, %	84,4 %	79,2 %	98 %
Recognition speed	0,6	0,5…1	0,5
Algorithm used	Neural networks	Hidden Markov Model	Hidden Markov Model
Algorithm used	Python	C/Java	C

5. Conclusion

Article analyzes features of voice commands pronunciation models.

Dependence: system accuracy on number of states for phonemes is considered; system accuracy on learning rate; system accuracy on value of training set.

As result, speech recognition system based on neural networks was developed.

Python was chosen as programming language, Kaldi framework was used for implementation. System was trained on data from TIMIT library. A recurrent neural network was chosen as architecture.

To implement developed system for speech recognition based on neural networks, block diagram of software was developed.

A speech recognition system is not easy to implement and requires an understanding of speech recognition basics.

In conclusion, developed system was compared with Speech Recognition from Google and PocketSphinx.

Implemented system achieved an accuracy of 84.4 %, which is very good result for similar system.

References

[1]. Sokolov, A., & Savchenko, A. V. (2019, January). Voice command recognition in intelligent systems using deep neural networks. In *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)* (pp. 113-116). IEEE. https://doi.org/10.1109/SAMI.2019.8782755

[2]. Muda, L., KM, B., & Elamvazuthi, I. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *Journal of Computing*, *2*(3), 138-143.

[3]. Li, N. (2020). An improved machine learning algorithm for text-voice conversion. *Journal of Intelligent & Fuzzy Systems, 40*(2), 2743-2753. https://doi.org/10.3233/JIFS-189316.

[4]. Ault, S. V., Perez, R. J., Kimble, C. A., & Wang, J. (2018). On speech recognition algorithms. *International Journal of Machine Learning and Computing*, *8*(6), 518-523. https://doi.org/10.18178/ijmlc.2018.8.6.739

[5]. Toda, T., Nakagiri, M., & Shikano, K. (2012). Statistical voice conversion techniques for body-conducted unvoiced speech enhancement. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(9), 2505-2517. https://doi.org/10.1109/TASL.2012.2205241

[6]. Desai, S., Black, A. W., Yegnanarayana, B., & Prahallad, K. (2010). Spectral mapping using artificial neural networks for voice conversion. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(5), 954-964. https://doi.org/10.1109/TASL.2010.2047683

[7]. Këpuska, V., & Bohouta, G. (2017). Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx). *Int. J. Eng. Res. Appl*, *7*(03), 20-24. https://doi.org/10.9790/9622-0703022024.

[8]. Ali, A. T., Eltayeb, E. B., & Abusail, E. A. A. (2017). Voice recognition based smart home control system. *International Journal of Engineering Inventions*, *6*(4), 01-05.

[9]. Barkani, F., Satori, H., Hamidi, M., Zealouk, O., & Laaidi, N. (2020). Comparative Evaluation of Speech Recognition Systems Based on Different Toolkits. In *Embedded Systems and Artificial Intelligence* (pp. 33-41). Springer, Singapore. https://doi.org/10.1007/978-981-15-0947-6_4

[10]. Mofrad, M. H., & Mosse, D. (2018, October). Speech recognition and voice separation for the internet of things. In *Proceedings of the 8th International Conference on the Internet of Things* (pp. 1-8). https://doi.org/10.1145/3277593.3277610

[11]. Saravanan, M., Selvababu, B., Jayan, A., Anand, A., & Raj, A. (2020, December). Arduino Based Voice Controlled Robot Vehicle. In *IOP Conference Series: Materials Science and Engineering* (Vol. 993, No. 1, p. 012125). IOP Publishing. https://doi.org/10.1088/1757-899X/993/1/012125

[12]. Dujmešić, N., Bajor, I., & Rožić, T. (2018). Warehouse processes improvement by pick by voice technology. *Tehnički vjesnik*, *25*(4), 1227-1233. https://doi.org/10.17559/TV-20160829152732

[13]. Reif, R., & Günthner, W. A. (2009). Pick-by-vision: augmented reality supported order picking. *The Visual Computer*, *25*(5), 461-467. https://doi.org/10.1007/s00371-009-0348-y

[14]. Niculescu, A., van Dijk, B., Nijholt, A., Li, H., & See, S. L. (2013). Making social robots more attractive: the effects of voice pitch, humor and empathy. *International journal of social robotics*, *5*(2), 171-191. https://doi.org/10.1007/s12369-012-0171-x

[15]. Tebelskis, J. (1995). *Speech recognition using neural networks* (Doctoral dissertation, Carnegie Mellon University).

[16]. Ahmad, M. A., Baker, J. H., Tvoroshenko, I., & Lyashenko, V. (2019). Computational complexity of the accessory function setting mechanism in fuzzy intellectual systems. *International Journal of Advanced Trends in Computer Science and Engineering*, *8*(5), 2370-2377. https://doi.org/10.30534/ijatcse/2019/77852019

[17]. Beck, M. W. (2018). NeuralNetTools: Visualization and Analysis Tools for Neural Networks. *Journal of Statistical Software*, *85*(11), 1-20. https://doi.org/10.18637/jss.v085.i11

[18]. Sewak, M., Karim, M. R., & Pujari, P. (2018). *Practical convolutional neural networks: implement advanced deep learning models using Python*. Packt Publishing Ltd.

[19]. Millstein, F. (2020). *Natural language processing with python: natural language processing using NLTK*. Frank Millstein.

[20]. Glackin, C., Wall, J., Chollet, G., Dugan, N., & Cannings, N. (2018, January). TIMIT and NTIMIT Phone Recognition Using Convolutional Neural Networks. In *International Conference on Pattern Recognition Applications and Methods* (pp. 89-100). Springer, Cham. https://doi.org/10.1007/978-3-030-05499-1_5

[21]. Swamy, S., & Ramakrishnan, K. V. (2013). An efficient speech recognition system. *Computer Science & Engineering*, *3*(4), 21. https://doi.org/10.5121/cseij.2013.3403.

[22]. Lipeika, A., Lipeikienė, J., & Telksnys, L. (2002). Development of isolated word speech recognition system. *Informatica*, *13*(1), 37-46. https://doi.org/10.3233/INF-2002-13103

[23]. Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M., & Rudnicky, A. I. (2006, May). Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* (Vol. 1, pp. I-I). IEEE. https://doi.org/10.1109/ICASSP.2006.1659988

[24]. Fayjie, A. R., Ramezani, A., Oualid, D., & Lee, D. J. (2017, July). Voice enabled smart drone control. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 119-121). IEEE. https://doi.org/10.1109/ICUFN.2017.7993759

[25]. Cutajar, M., Gatt, E., Grech, I., Casha, O., & Micallef, J. (2013). Comparative study of automatic speech recognition techniques. *IET Signal Processing*, *7*(1), 25-46. https://doi.org/ 10.1049/iet-spr.2012.0151