UDC 004.75

Oleksandr Verba, Yurii Vynohradov, Youssef El Ajjad.

MODIFICATION OF THE SPEECH RECOGNITION ALGORITHM FOR VOICE-USER INTERFACE

The paper deals with the speech recognition algorithm for voice user interface which is optimized under realization on FPGA.

Keywords: voice-user interface, hidden Markovskie models, Field-Programmable Gate Array

Relevance of the research topic. Voice Interfaces (VUI) are an evolution of interaction that makes it easy to enter or receive information. Creating interfaces that support and offer more ergonomic and natural forms of human - machine dialogue predetermined by the introduction of information technology in professional and everyday human activities. Therefore, the development of specialized voice interface tools based on speech recognition technologies is an urgent scientific and practical task.

Formulation of the problem. Digital processing of speech signals is the basis for building voice interfaces. The article proposes a modification of the algorithm for the method of hidden Markov models in speech recognition for the voice interface, which is optimized for implementation on FPGA.

Analysis of recent research and publications. Currently, the following areas of research and development can be distinguished in the field of improving performance and implementing independent speech recognition modules for the voice interface:

– firstly, the introduction of hardware support for algorithms preliminary processing and selection of signs (for example, the implementation in programmable logic circuits (FPGA) of a block for finding Mel Frequency Cepstral Coefficient) [1, 2];

– secondly, hardware implementation of recognition algorithms.

The latter direction is represented by many works. At the same time, there is a noticeable general trend in the development of hardware implementations of the recognition unit:

1) user-programmable logic is used as microcircuits because of their availability and versatility;

2) in the implementation of speech recognition devices, the main focus is on the introduction of hardware support for hidden Markov model algorithms - the forward motion algorithm and the Viterbi algorithm. As a prerequisite for this, the high computational complexity of the indicated algorithms is indicated.

Highlighting unexplored parts of a common problem. To solve the problems of hardware implementation of speech recognition algorithms, both hardware methods are proposed (for example, in [3] it is proposed to construct a systolic matrix for performing calculations using forward and Viterbi algorithms) and algorithmic ones. For the development of independent modules for the tasks of the voice interface, the use of a small dictionary is characteristic, which distinguishes this problem from the general approach and speech recognition methods.

Object of study. The purpose of this article is to develop a modified speech recognition algorithm based on hidden Markov models with less computational complexity, which is optimized for implementation on FPGA.

Statement of the main material. Discrete Markov chains describe random processes that occur in discrete time and at each moment reside in one of $i \in I$ state, the set of states I is finite or countable. The transition from a state to a different state is random and is determined by the transition matrix A = {aij}, where aij is the transition probability from state i to j. The initial state is also randomly selected according to the vector aij of the initial distribution π = {πi} where πi is the probability that the process is in state i at the initial moment.

During its evolution, the Markov process passes through a sequence of states Q = q1, q2, ... qT in T cycles. The probability that at times 1,2, ..., T these states turned out to be i1, i2, ..., iT is determined as follows:

$$P(q_1=i_1,\ q_2 = i_2,\ldots,q_T=i_T)\ =\ \pi q_1 i_2,\ldots,q_T=i_T) \qquad (1)$$

In this case, the transition to the current state is determined only by the previous state.

$$Pq_i|Q_1^{i-1} = Pq_i|q_{i-1} \qquad (2)$$

where Q1i-1 = q1, q2, ..., qi-1. This feature of Markov chains is called the Markov property or Markov assumption, and the process itself is characterized as a “memoryless process” [3].

This model can be more complicated by separating the states and observed events in such a way that the occurrence of an event in each state will also be probabilistic. The result is a double stochastic process with a hidden layer - a random sequence of states, and an external layer of observed random output values. Such a model is called the hidden Markov model (HMM).

For HMM, in addition to the set of states I, it is necessary to introduce a finite set of observable values of C (alphabet) . During its operation, the HMM emits a chain of observed values $O = (o_1,o_2,\ldots,o_T)$, $o \in C$

Thus, a discrete hidden Markov model is determined using:

1. $C = \{c_1,c_2,\ldots,c_M\}$ - the alphabet of the observed values;
2. $I = \{i_1,i_2,\ldots i_N\}$ - sets of states accepted by the system;
3. $A=\{a_{ij}\}$ - transition probability matrices;

4. B= {$b_i(c_k)$} – matrix of output probabilities, where $b_i(c_k)$ is the probability of observing the symbol $c_k \in C$ when the model is in state i;

5. π = {π_i} – probability vectors of the initial state.

A hidden Markov model can be defined as a triple λ = (A,B, π).

The use of HMM in speech recognition is based on the construction of phonemes stochastic models, words and entire phrases. The choice of a particular language object depends on the tasks that the developed speech recognition system should solve. The chain of feature vectors plays the role of the observed sequence. If the feature vector is a continuous value (for example, a set of Mel-frequency cepstrum coefficients), then a mixture of Gaussian probability densities is used to model it. In this work, the obtained Mel-frequency cepstrum are quantized, which allows the use of discrete one-dimensional $b_i (c_k)$.

To use hidden Markov models in speech recognition, it is necessary to solve the following three problems [2]:

1. The assessment task - the model λ and the output sequence O are given. To find the probability P (O | λ) which means, determine the probability that the model λ generated the sequence O.

2. Decoding task - a model λ and an output sequence O are given. To find the most likely sequence of states Q that could spawn O.

3. Learning objective - the model and the training sequence O are given. Choose the parameters of the model λ in such a way as to maximize the probability P (Q | λ).

The use of HMM for recognition of isolated words is based on the calculation of the direct probability distribution function $ά_{tj}$ which is defined as the probability of observing the sequence $O_t = (o_1,o_2,\ldots o_t)$, being in state j at time t on the model λ = (A, B, π) [5]:

Obviously, that the calculation of $ά_t(j)$ is recursive. To increase efficiency, the recursion can be converted to a loop. By reaching the end of the observed sequence, which means until t = T, we need to add $ά_T j$ for all states, obtaining the probability of observing the sequence $O = (o1,o2,\ldots,o_T)$ for a given HMM λ:

$$P(O \mid \lambda) = \sum_{j=1}^{m} ά_T(j) \quad (3)$$

This probability can be used in the recognition of isolated words: each word is modeled by HMM, and when recognizing a word, it is necessary to choose the HMM that is most likely to generate the observed sequence O.

$$w^* = \text{argmax}w = WP(O|)\lambda_W \quad (4)$$

When HMM training, it is necessary to evaluate its parameters λ = A, B, π to maximize the probability of observing the training sequence P (O | λ).

To eliminate the decrease in Pt (O | λ), there are two tricks: scaling and replacing the value ά t (j) with its logarithm. The first method introduces a scaling factor st such that $\sum_{j=1}^{m}$ ά T (j) $=$ 1:

$$\mathrm{st} = 1/\sum_{\mathrm{j=1}}^{\mathrm{m}} ά \, \mathrm{t} \, (\mathrm{j}) \qquad (5)$$

Multiplying ά t j by st every step t, at t = T, the product of scaling factors accumulates S(T) = $\sum_{j=1}^{m}$ st Final probability P (O | λ)= P_t (O|λ) fully expressed in terms of S (T):

$$P(O|\lambda) = 1 / S(T) \qquad (6)$$

The advantage of using scaling factors is that you can go to the logarithm of P (O | λ), which decreases much more slowly.

$$\log P(O|\lambda) = -\sum_{\mathrm{t-1}}^{\mathrm{T}} \mathrm{st} \qquad (7)$$

In addition to eliminating the exponential decrease of P (O | λ), the use of the logarithm allow to replace the multiplication operations by additions, which positively affects the algorithm speed.

If we introduce the logarithm a t (j) and do all the calculations only in the logarithmic representation is:

$$\log ά_1(j) = \log \pi_j + \log b_j (o_1) \qquad (8)$$

$$\log ά_1(j) = \log bj \; (ot) + \log \left(\sum_{i=1}^{N} a_{ij} \alpha_{\mathrm{t-1}}(i)\right. \qquad (9)$$

To calculate the sum logarithm where the pairs a_{ij} ά $_{t-1}$(i) and a_{i+1j} ά $_{t-1}$(i+1) are P1 and P2:

$$\log_b (P_1+P_2) = \log_b P_1 + \log_b (1+b^{\log bP2 - \log bP1}) \qquad (10)$$

The essence of the modification of the algorithm, based on the simplification of the calculation of the sum of the logarithms in (9). We transform the formula (9), thus:

$$\log_b(P_1 + P_2) = \log_b P_1 + \log_b(1 + P_2 / P_1) = \log_b P_2 + \log_b(1 + P_1 / P_2) \qquad (11)$$

If $P_1 > P_2$ then $\log_b(P_1 + P_2) \approx \log_b P_1$ And vice versa: if $P_2 > P_1$ then $\log_b(P_1 + P_2) \approx \log_b P_2$. Then you can replace the exact calculation with an approximate one:

$$\log_b(P_1 + P_2) \approx \log_b (\max(P_1,P_2)) \qquad (12)$$

Considering, that the logarithm is a monotonically increasing function, we obtain:

$$\log_b(P_1 + P_2) \approx \max(\log_b P_1 , \log_b P_2) \qquad (13)$$

In this case, for ά $_t$(j) :

$$\log ά_t(j) = \log_b (o_t) + \log \left(\sum_{i=1}^{N} a_{ij}\alpha_t (i) \right) \approx$$
$$\approx \log_b (o_t) + \max(\log ά_{ij} + \log ά_{t-1j} (i)) \qquad (14)$$

Forward Algorithm Completion which is the calculation of P (O | λ), can be replaced by the calculation $\sum_{i=1}^{N} \log a_t(j)$ of the model λ.

In the proposed modification, multiplication operations are completely absent and only addition is used.

To check the adequacy of the model, the Matlab package was used possessing a huge library of mathematical functions and means of visualizing the results. In the model for speech recognition, it is necessary to prepare the initial data for the organization of its comprehensive testing. These initial data represent a speech base that includes a sufficient number of training and test cases.

The recognition result was evaluated using traditional metric classification tasks: Precision, Recall, and F1 metric. Let w * (x) be the class predicted by the classifier for pronouncing x ∊ X (where X is the set of test cases), and y (x) be the true class of x, so that w *: X → W and y: X → W. To evaluate the classification results, a confusion matrix K = k_{ij} is constructed for i, j = 1, ..., Ξ, in which each element represents the number of predictions that x belongs to class i, while its true class is j:

$$k_{ij} = \sum_{l=1}^{M} ((w * (x_l) = i) (\xi (y(x_l) = j) \qquad (15)$$

The indicator function is used in formula (15)

$$\xi e = 1 \text{ if e is true}$$

$$\xi e = 0 \text{ if e is false}$$

Thus, the diagonal elements of the error matrix contain the number of correct predictions for each class, while the remaining elements contain the number of erroneous predictions. It is convenient to normalize the error matrix by the number of test cases for each class, so k_{ij} would be the conditional probability of predicting the class w * x_l = i, provided that the true class x is y x = j:

$$k_{ij} = P(w^* (x) = i| y (x) = j) = k_{ij} / m \qquad (16)$$

The diagonal values of the normalized error matrix contain the values of the completeness metric R_i for each class i. The completeness metric R_i is the probability of a correct prediction, provided that the true class x is i:

$$R_i = P(w^* (x) = i| y (x) = j) = k_{ii} \qquad (17)$$

Another metric for evaluating classifier performance is P_i accuracy for class i. The accuracy of P_i is the probability that the true class x is i, provided that the class w * (x) = i was predicted:

$$P_i = k_{ii} / \sum_{\Xi}^{k_y} k_{ij} \qquad (18)$$

Finally, F_1 (i), the metric for each class, is a combination of accuracy and completeness metrics that can be interpreted as their weighted average:

$$F_1(i) = 2\cdot (Pi\cdot Ri / Pi +Ri) \qquad (19)$$

The metrics described above provide an exhaustive assessment of the performance of the classifier. For each recognition method implemented in software and in the Automated machine learning software model, the metrics described above and their average values in the Table 1 were calculated.

As can be seen from the graph in Figure 1., although the recognition accuracy decreases with increasing dictionary size, it remains at the level of 0.9510 for a dictionary of 100 words. Usually, with a further increase in the recognition dictionary, accuracy only decreases. However, it should be noted that a dictionary of 100 words is sufficient for many voice recognition systems.

Table 1.

Calculation of the metrics and their average values

Recognition method	**Completeness metric**	**Accuracy metric**	**F_1 - metric**
Modified Algorithm P(O | λ)	0.7820	0.8202	0.7790
Arbitrary Sound Algorithm	0.9749	0.9510	0.9460

Findings. The article analyzes the ways and methods of creating a voice interface, speech recognition methods are considered and the apparatus of hidden Markov models is selected.

A method for implementing the algorithm based on the representation of probability using the flow intensity and approximate calculations using logarithms is developed, which made it possible to reduce its complexity.

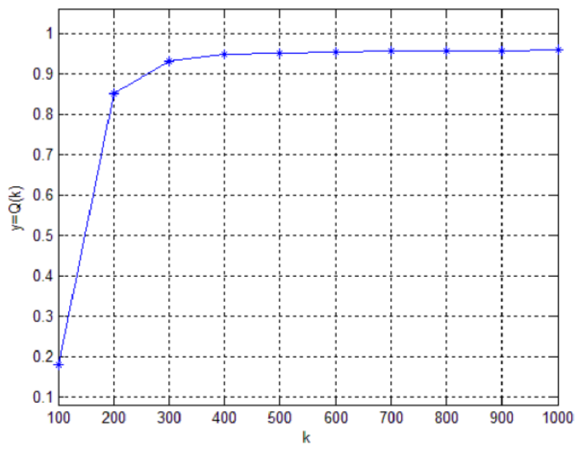

Fig. 1. The accuracy of word recognition.

On a fixed speech base, the proposed implementation of speech recognition was investigated and the developed methods were evaluated. Dependencies of accuracy on various parameters of the selected length and sequence of words are obtained.

Literature

1. V. V. Ngoc, J. Whittington, J. Devlin, Real-time Hardware Feature Extraction with Embedded Signal Enhancement for Automatic Speech Recognition // Speech Technologies, Intech, 2011. – Pp. 29-54.

2. M. Bahoura, H. Ezzaidi, Hardware implementation of MFCC feature extraction for respiratory sounds analysis // 8th Workshop on Systems, Signal Processing and their Applications, 2013. – Pp. 226-229.

3. Mosleh M., Setayeshi S., Mehdi Lotfinejad M., Mirshekari A., FPGA implementation of a linear systolic array for speech recognition based on HMM // The 2nd International Conference on Computer and Automation Engineering (ICCAE), 2010. Vol. 3, – pp. 75-78.

4. Rabiner L., Juang B.-H. Fundamentals of speech recognition – Prentice Hall. – 1993, 507 p.

5. Бондаренко Л. В., Вербицкая Л. А., Гордина М. В., Основы общей фонетики. – М.: Академия, 2004. – 160 с.

AUTHORS

Oleksandr Verba (supervisor) – associate professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: olverba@gmail.com

Yurii Vynohradov (supervisor) – Senior Lecturer, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: vinograd514kpi@gmail.com

El Ajjad Youssef – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: ucefelajjad@gmail.com