Foreign Language Skills For Professional Improvement With Learning Trajectories

Wang Lina[1], Liu Juanyin[1], Li rui[2], Ma Jifei[3*]
[1]Hunan Industrial Vocational and Technical College
Changsha, 410000, Hunan, China

[2]Jiangxi Industrial Vocational and Technical College
Nanchang, 330000, Jiangxi, China

[3]Anhui Normal University, Wuhu, 241000, Anhui, China
pieji68954lianlia@163.com

ABSTRACT: *With the improvement of China's international status, the demand for foreign language application talents is becoming increasingly significant for hosting the Winter Olympics. To meet this demand, we propose a solution based on the Viterbi algorithm aimed at cultivating applied talents with foreign language skills and professional knowledge required for the Winter Olympics. This plan first determines the foreign language skills and professional knowledge required for the Winter Olympics, including English, French, German and other languages, and professional knowledge related to ice and snow sports. Then, we used the Viterbi algorithm to identify students' learning patterns and difficulties based on their learning trajectories and performance data, providing them with more personalized and targeted teaching solutions. In addition, we have combined the cultivation of professional knowledge and foreign language skills in ice and snow sports to enhance students' practical abilities and comprehensive qualities. Through experimental verification, we found that the Viterbi algorithm-based solution can significantly improve students' foreign language application ability and professional knowledge level. This program has higher teaching efficiency and lower teaching costs than traditional training models. In addition, students also showed higher satisfaction and stronger learning motivation towards the program.*

Keywords: Viterbi Algorithm, Application-oriented Talents of Foreign Languages, Train

Received: 2 March 2023, Revised 139 June 2023, Accepted 25 June 2023

DOI: 10.6025/jcl/2023/14/3/82-89

1. Introduction

With the deepening of internationalization, the demand for application-oriented talents in foreign languages gradually increases. China is about to hold the Olympic Winter Games in 2022, so the training of application-oriented talents of foreign languages should be prepared [1]. This paper has studied the training schemes of application-oriented foreign language talents

based on the Viterbi algorithm. This paper has built a speech recognition system based on the Viterbi algorithm, and the aim is to promote the training of application-oriented foreign language talents in the 2022 Winter Olympics. In recent years, with the arrival of the mobile internet and the big data wave, various intelligent terminals have appeared in the daily lives of human beings, and voice recognition technology, which has come with intelligent terminals, is beginning to change the way we live and work, and it is gradually becoming a kind of the primary human-computer interaction mode [2]. In the most recent applications of human-computer interaction, voice recognition technology has played a very important role, and it is rapidly developing into one of the key technologies for transforming the future of human life [3]. The use of voice recognition technology to train application-oriented foreign language talents is worth exploring deeply [4].

2. State of the Art

In recent years, Voice recognition technology is beginning to change how we live and work, becoming a primary man-machine interactive mode. The opening of this trend is attributed to progress in several key areas [5]. Firstly, Moore's law continues to play a role. Using multi-core processors 0 universal image processing unit and CPU/GPU aggregate, today's computing power has increased by several orders of magnitude compared with a decade ago. All this makes it possible to train more complex models, and these models, which have more computational requirements, have significantly reduced the error rate of automatic speech recognition systems. Secondly, compared with the former, we can get more data now, and this is thanks to the continued rapid growth of the Internet and cloud computing. By building a model that is based on massive data collected from real application scenarios, we can eliminate many of the previous assumptions about the model and make the system more robust. Thirdly, mobile devices, wearable devices, smart home appliances and in-vehicle infotainment become popular. On these devices and systems, alternative interactive modes such as keyboard and mouse are far less convenient than personal computers [6]. Voice is a natural way to communicate with each other. And at the same time, it's a skill most people already have. Therefore, on these devices and systems, voice can be a better way to interact [7].

3. Methodology

3.1 Build Voice Command Model

This design mainly focuses on recognising voice commands, so we will complete the construction of the hidden Markov model network model of all instructions. There are two specific jobs to be done: One is to import the hidden Markov model of each trained phoneme into memory and store it well with a certain data structure to prepare the hidden Markov network of all instructions; Two is for each voice command, using the Connect operation, the phonemic hidden Markov model which corresponds instructions links to the hidden Markov chain which is in alphabetic order of instructions. Then the Union operation splices the hidden Markov chain of all instructions and constructs a hidden Markov network containing all identification instructions. Finally, the static model is added before and after the network, and the whole system is built with the hidden Markov network model. Analyzing the data of the phonemic hidden Markov model we trained, We found that there are three different types of data: One is the state transfer matrix of the data of the phonemic hidden Markov model, and it includes the name, the number of the ranks and the concrete data which are about the state-transition matrix; The second is that the Gaussian mixture model data corresponds to each status in the data of the phonemic hidden Markov model, and it includes the name of state, the number of Gaussian mixtures and the specific data for each Gaussian mixture; Three is the phonemic hidden Markov model, and it includes the name of the phonemic hidden Markov model, the number of states and the Gaussian mixture model of the observed sequence of each state output. Meanwhile, it also includes transfer matrix data for all states.

Corresponding to the previous data analysis, the defined structure TRANSFER saves the transfer matrix of the data about the phonemic hidden Markov model, and it is shown in Table 1; GAUSSIAN and STATE save Gaussian mixture model data which corresponds to each status in the data of the phonemic hidden Markov model. We also built tables. For the limitation of the thesis, I will only give the necessary details here. Afterwards, we also built HMM to save the phonemic hidden Markov model.

The process of storing the entire data store is encapsulated in the load Hmm (Hmm FP) function, and hmm *Fp* is the file pointer of the phonemic hidden Markov model, which we got in the foregoing training module. We can define three numbers *T, S* and *H,* to find the state-transition matrix0, status, and hidden Markov model data. And then, we define load, *Hmm* to complete the data storage. Afterwards, we find the flag in the data and store all state transfer matrixes about the data of the phonemic hidden Markov model. We use the find T function to find the flag; every '~t' flag represents a transfer matrix. Then, the data of each transfer matrix is stored in the TRANSFER node in turn. Meanwhile, we linked all of the TRANSFER nodes to a linked list and gave the head node Transfer Head of the transfer matrix linked lists. We find the '~s' flag in the data and store the Gaussian

Specific members	Significance
char *transfer Name	The name of the transfer matrix
int size	Row and column number of the transfer matrix
double **transfer data	Specific data of the transfer matrix
struct TRANSFER *next	Point to the next transfer matrix node

Table 1. TRANSFER Specific Members and Meaning-Specific Members

mixture model data, which corresponds to each status in the data of the phonemic hidden Markov model. Then, we use the find S function to find the '~s' flag, and every '~s' flag represents the state of a hidden Markov model. The data of each state list is stored in the STATE node in turn. Meanwhile, we linked all the STATE nodes to a linked list and gave the head node State Head of the state list. We find the '~h' flag in the data and store the data of the phonemic hidden Markov model. Then, we use the find H function to find the flag, and every '~h' flag represents one phonemic hidden Markov model. The data of each phonemic hidden Markov model is stored in the HMM node in turn. Meanwhile, we linked all the HMM nodes to a linked list and gave the head node Hmm Head of the HMM model data linked list.

Then, we construct the hidden Markov network of all instructions. Firstly, we analyze the construct procedure of the single instruction about the hidden Markov chain and take "Please turn it off" as an example. The "Please turn it off" instructions have six phonemes, and we define Connect operate. According to the phonetic structure of "Please turn it off", the six phonemic hidden Markov models of *v-q+ing0c-ing+g0ing-g+uan0g-uan+j0uan-j+I* and *j-i+sil* are concatenated together in turn to construct a hidden Markov chain which represents " Please turn it off", and it is shown in the following figure 1. Two hidden Markov models are connected. The previous hidden Markov model deleted the last state, and a hidden Markov model is deleted from the headmost state.

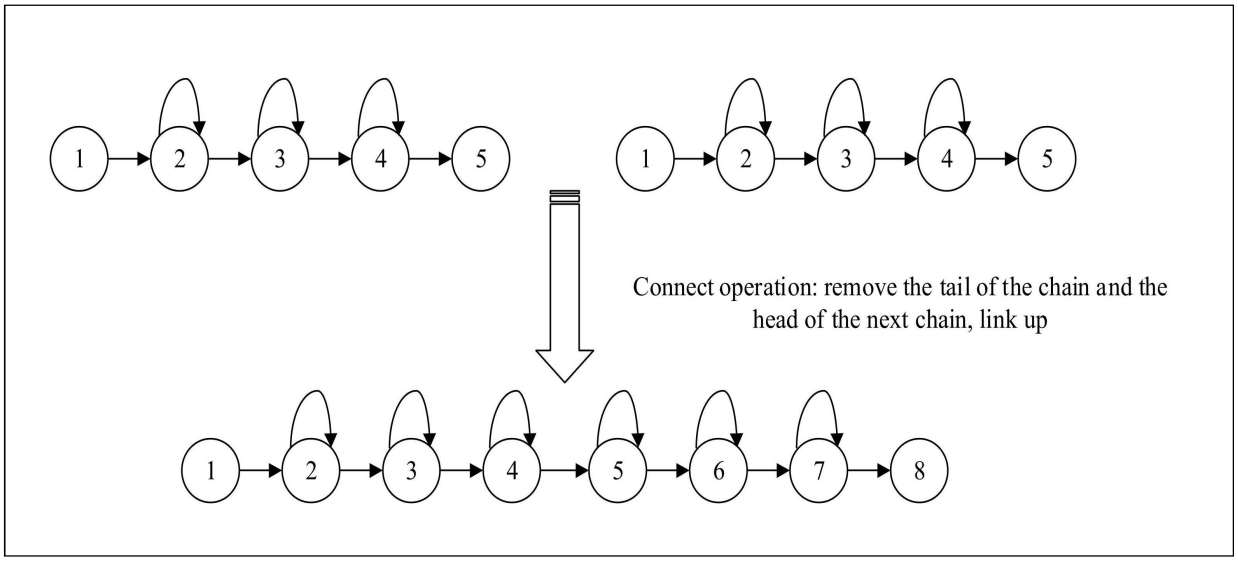

Figure 1. Connect operation

Then, define the Union operation; the hidden Markov chain of each instruction is connected to a hidden Markov network. At the same time, the start and end of the network take advantage of the Connect operation to add the silent, hidden Markov chain.

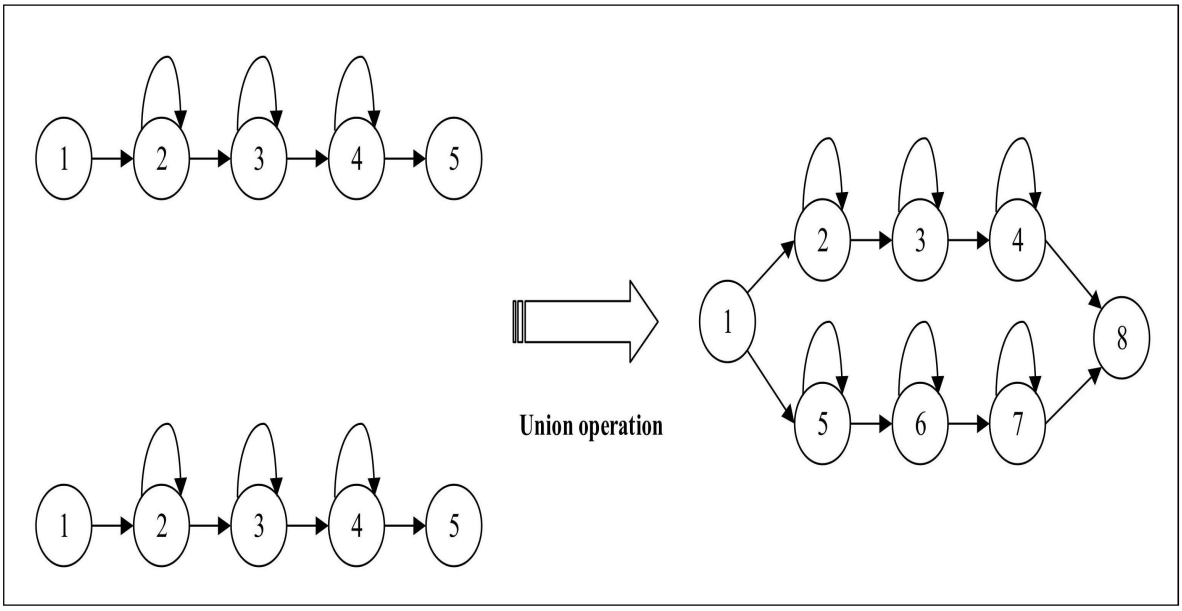

Figure 2. Union operation

Finally, based on the above analysis, we define the structure NODE to store the state node of the hidden Markov, LINK stores the Markov chain, and NET stores the hidden Markov network. We encapsulate a single phonemic hidden Markov chain in a function hmm To Net. Connect and Union operations are separately encapsulated in Connect Net and Union Net. The entire network build is encapsulated in b Main Net, and the release of network models is encapsulated in a function-free Memory. Using hmm To Net, the STATE of each phoneme is linked to the hidden Markov chain representing the phoneme according to the hidden Markov data rule. Then, we use connect Net to link the hidden Markov chain of each instruction's phonemes to constitute a hidden Markov chain representing the instruction. Use the union Net function to link each instruction's hidden Markov chain side by side to constitute a hidden Markov chain representing all instructions. Afterwards, we use the Loop Print function to add rotation to the hidden Markov network. It's just to add the rotation; the whole network is wholly constructed. Finally, use the add Keyid function to add the identification flag Keyid for each network instruction, and Keyid is the identification result of decoding later. Therefore, it is very critical.

3.2. Viterbi Algorithm Analysis

The token passing algorithm is an improved Viterbi frame synchronization decoding algorithm, and it is very suitable for searching the hidden Markov model structure without jumping from left to right. When we search for the hidden Markov network model, there is a probability between different states of jumping, and there is also a probability of the rotation of the same state. Meanwhile, there is also a value of the observed sequence probability of the corresponding state. These values are very useful for decoding, so we have to save them to define a Token to store the value of the previous cumulative probability. At the beginning, we initialize a Token for all possible state nodes. With time, when we process a frame of voice signal each time, each Token is propagated along the transferred arc, which is connected to it in the network and until the next one, which has the state of the output probability density function. Finally, the exit status of the hidden Markov network is reached. When a state node has many successive state nodes, that Token will propagate along all paths, making the Token go through all possible paths.

When taken passes between different states, the cumulative value of the probability of the newly generated token should be combined with the probability value of the state jump and the probability value of the observed sequence corresponding to the new state. However, the status could also rotate, and the token should be updated now. The probability value of the stored value in the token is added to the probability value of the state rotation and the probability value of the observed sequence corresponding to the current state. Meanwhile, when Token migrates in the network, its path and voice commands must be recorded and the degree of detailed path reservation depends on the need to identify the output. In this design, we mainly record logarithmic likelihood and voice command identification. Finally, we find the Token, the most logarithmic value in the output

node. According to its corresponding history, it can be recorded to find all the state nodes and the transfer arc this Token experienced. At the same time, the recognition result can also be obtained by using the voice command identity in the Token data structure. For some state *i* at time *t*, we might get multiple Tokens passed from *t*–1. In that way, how do we calculate the logarithm probability value in the token of state *j*? When we calculate actually, we'll copy each token of the passing state *i* to the adjacent state *j* connected to it. The value of formula (1) increases the logarithmic probability value of the token and the status will update the token at every moment.

$$\log[a_{ij}] + \log[b_j(o(t))] \tag{1}$$

The implementation steps of the token transfer algorithm are as follows. The first is initialization and setting a token in the initial state of the hidden Markov network model. In this taken, the initial probability is 0, and the initial probability value of the setting token for all other states is ∞. The second is circular iteration. Starting from the $t = 1$, the characteristic vector of each frame signal is processed until the last frame signal. In the state jump process, the token of the current state will be copied to the new state, and the cumulative value of the new status token needs to be updated. Assuming the current state is *I* and the new state is *j*, and the calculation formula for the probabilistic cumulative value of the new token is shown in 2:

$$p_j(t) = p_i(t) + \log[a_{ij}] + \log[b_j(o(t))] \tag{2}$$

If it revolves on its axis, the calculation process is similar to formula (2). If the state *i* rotates, the cumulative value of the probability of the latest token of state *i* is just replaced by the formula (2) *j* for *i*. Each state only retains the token with the maximum cumulative value; terminate the Token that checks all terminating states of the network model and the Token that has the maximum cumulative value that are we need and we can receive the result according to this Token.

4. Result Analysis and Discussion

In this paper, an experimental speech test sample set has been chosen from the 14 test instructions and the contents are shown in Table 2. The voice test sample collected 20 people, and the ratio of men to women is 1:1. At the same time, speakers have standard Chinese Mandarin. Each speaker has a normal speed (about 240 syllables per minute) to read 14 test instructions, and the interval between the two instructions is more than one second. Finally, there are 280 voice test samples. The surrounding environment is quiet when we record, and there is no greater background noise. Voice test samples are named successively after their instructions, such as the first voice command that is "Please turn off the machine", it is called shutdown 1.

Please turn off the machine	Please open the machine	Please play video	Please open the address book	Please open the Notepad
Please take a picture	Please turn off the music	Please play music	Please open the album	Please turn off the ideo
Please turn off the alarm clock	Please lock the screen	Please check the weather	Please open the browser	

Table 2. Speech Test Instructions

To test the noise resistance of the system, we add noise to each test speech sample during the test, and the noise used is additive white noise. The test process adds up to four proportions of noise, and they respective are the white noise of 15dB, 20dB, 25dB and 30dB and then repeat the above process. Table 3 lists the recognition accuracy of all voice commands in different SNR environments and figure 3 shows the system recognition rate curve under different SNR environments.

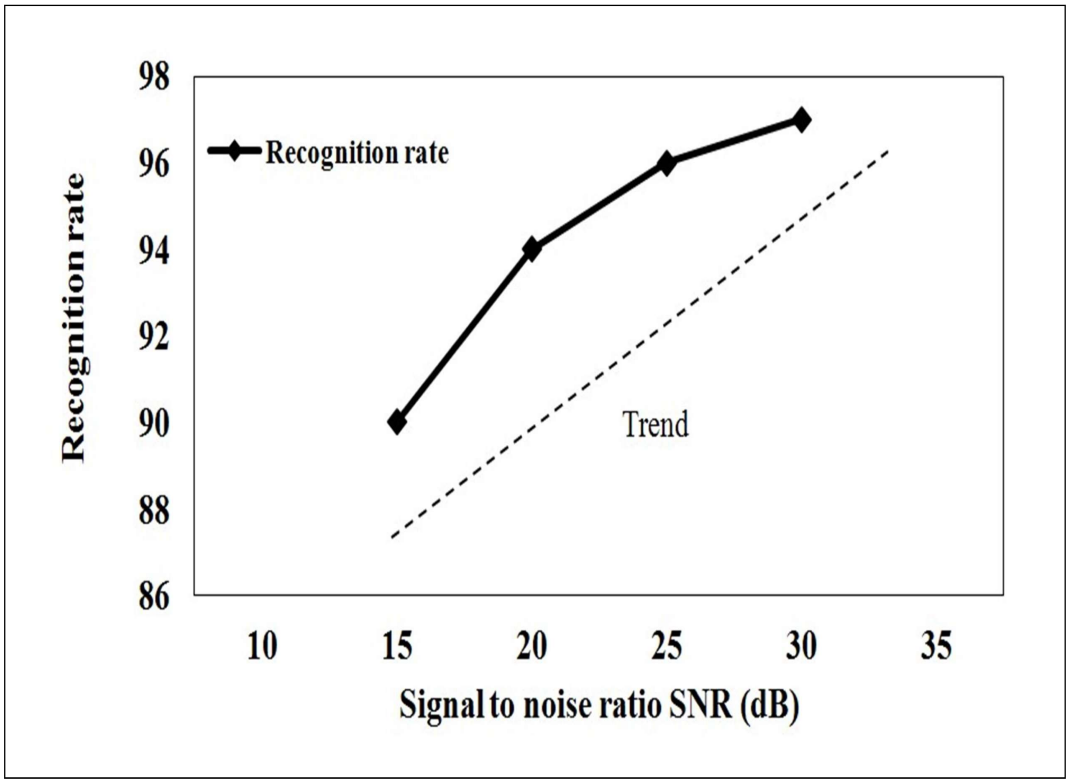

Figure 3. System recognition rate curve in different signal to noise ratio environment

Signal-to-noise ratio	Correct recognition of the number of voices	Total test speech number	Recognition rate
15dB	252	280	90.0%
20dB	264	280	94.2%
25dB	269	280	96.1%
30dB	274	280	97.8%

Table 3. Comparison of Recognition Rates of Different SNR Noises in Each System

Though Figure 3 and Table 3 can be found, as the noise is enhanced, the signal-to-noise ratio is getting smaller, and the recognition rate of the voice recognition system is decreasing gradually. However, the rate of recognition rate of this system could be faster. Between the signal-to-noise ratio of 20d B to 30d B, this system's recognition rate is very low; Under 20d B, the system recognition rate declined rapidly. In general, the anti-noise system is stronger. The cause of this phenomenon is mainly the training samples of the system, which are noisy, making our training model more anti-noise. This also tells us that selecting voice samples with noise during training can improve the recognition rate of speech recognition systems 0interference immunity, and robustness. Therefore, we have to pay attention to it in actual product development.

And then in the decoding process, we discard many tokens much smaller than the maximum token probability. Then, we define a Beam, and the Beam equals the cumulative value of the maximum token probability divided by the probability of the current

token. Through the experiment, the value of Beam is discussed. We calculate the recognition rate and real-time factor under the Beam. Figure 4 shows the corresponding curve of the system's recognition rate and real-time factor under different beams.

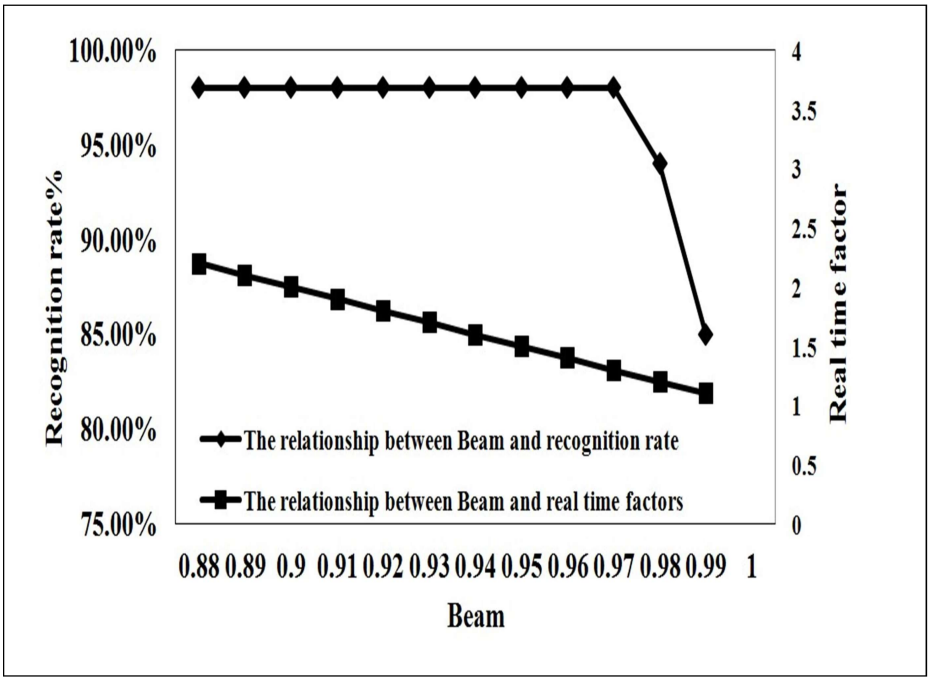

Figure 4. The curve of the recognition rate and the real-time factor of the system under different Beam

It can be inferred from Figure 4 that as the value of Beam increases, the recognition rate has remained constant. When the Beam value reaches 0.97, the recognition rate drops sharply if the value of the Beam continues to increase. The real-time factor keeps getting smaller as the value of Beam increases, which means that the identified speed becomes faster and faster as the value of Beam increases. Considering the above data, if the system has a higher recognition rate, we can let Beam be 0.97, the corresponding recognition rate is 98.92%, and the real-time factor is 0.38; if the system does not require that high recognition rate and the speed requirement is higher, we can let Beam be 0.98, and the corresponding recognition rate is 94.29%, and real-time factor is 0.22. Real-time factors 0.22 and 0.38 are not different; considering the recognition rate and real-time factor, it is appropriate to select Beam for 0.97. Table 4 shows the result comparison of Beam for the 0.97 pruning and the initial token transfer algorithm.

Type of test	**Correct recognition number**	**Test sample number**	**Average recognition rate**	**Mean real time factor**
Token passing	277	280	98.92%	3.01
Prune	277	280	98.92%	0.38

Table 4. Comparison of Test Results

We can see from Table 4, that the pruning optimization decoding can significantly improve the decoding speed, and the decoding speed has increased eightfold. Compared to the original token passing algorithm, its recognition rate remained constant and maintained at a high level. Therefore, we adopted the pruning decoding of Beam for 0.97 in the final system, significantly improved the decoding rate and ensured the recognition rate.

5. Conclusion

Voice recognition technology, which comes with smart terminals, begins to change how we live and work, gradually becoming a main man-machine interactive mode. And the use of voice recognition technology for the application-oriented talents of foreign languages is worth exploring deeply. China will host the Winter Olympics in 2022; as the date approaches, China gradually began to train and select the application-oriented talents of foreign languages. Computer technology is developing rapidly, and artificial intelligence technology is very popular. Hence, using computer technology to train the application-oriented talents of foreign languages is worth exploring. This paper has studied the training of the application-oriented talents of foreign languages based on the Viterbi algorithm. Firstly, based on the actual training of the application-oriented talents of foreign languages, the voice instruction model has been constructed by combining related theories in this paper. Then, this paper explores the algorithm on this model, and an optimized Viterbi algorithm is constructed. Finally, the experiment has been carried out, the improved algorithm is tested, and the optimization of the algorithm is effective.

Acknowledgement

Soft Science Research of Hebei Science and Technology Project (No. 16455325) financed the work presented in this paper.

References

[1] Enns, R. and Morrell, D. (1995). Terrain-aided navigation using the Viterbi algorithm. *Journal of Guidance Control & Dynamics*, 18 (6), 1444-1449.

[2] Yao, X., Zhu, D., Ye, S., Yun, W., Zhang, N. and Li, L. (2016). A field survey system for land consolidation based on 3S and speech recognition technology. *Computers & Electronics in Agriculture*, 127, 659-668.

[3] Chen, L., Mao, X. and Yan, H. (2016). Text-Independent Phoneme Segmentation Combining EGG and Speech Data. *IEEE/ACM Transactions on Audio Speech & Language Processing*, 24 (6), 1029-1037.

[4] Kermani, M.M., Singh, V. and Azarderakhsh, R. (2016). Reliable Low-Latency Viterbi Algorithm Architectures Benchmarked on ASIC and FPGA. *IEEE Transactions on Circuits & Systems I Regular Papers*, 7 (9), 1-9.

[5] Lee, Y., Kim, Y., Lee, Y. et al (2016). Development of Application Using the Voice Recognition Technology for Improving Speech Production Skills of Children with Speech Sound Disorders. *IEEE Transactions on Circuits & Systems I Regular Papers*, 2016, 4 (2), 77.

[6] Lu, T., Xu, Z. and Huang, B. (2017). An Event-Based Nonintrusive Load Monitoring Approach: Using the Simplified Viterbi Algorithm. *IEEE Pervasive Computing*, 16 (4), 54-61.

[7] Hanif, M.K. and Zimmermann, K.H. (2017). *Accelerating Viterbi algorithm on graphics processing units. Computing*, 2 (3), 1-19.