Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 2312908, 9 pages
https://doi.org/10.1155/2020/2312908

Research Article

A Novel Method for Recognizing Vietnamese Voice Commands on Smartphones with Support Vector Machine and Convolutional Neural Networks

Quang H. Nguyen and Tuan-Dung Cao

School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 10000, Vietnam

Correspondence should be addressed to Tuan-Dung Cao; dungct@soict.hust.edu.vn

Received 9 October 2019; Revised 5 December 2019; Accepted 13 February 2020; Published 1 March 2020

Academic Editor: Manuel Fernandez-Veiga

This paper will present a new method of identifying Vietnamese voice commands using Google speech recognition (GSR) service results. The problem is that the percentage of correct identifications of Vietnamese voice commands in the Google system is not high. We propose a supervised machine-learning approach to address cases in which Google incorrectly identifies voice commands. First, we build a voice command dataset that includes hypotheses of GSR for each corresponding voice command. Next, we propose a correction system using support vector machine (SVM) and convolutional neural network (CNN) models. The results show that the correction system reduces errors in recognizing Vietnamese voice commands from 35.06% to 7.08% using the SVM model and 5.15% using the CNN model.

1. Introduction

Users' need for devices that support voice communication is increasing rapidly. Voice is the natural approach to human-machine communication, so voice control support systems must meet the needs of a variety of users. According to Jeffs' survey [1] of US smartphone users in 2019, the number of users who use their voices to communicate with their phones has increased dramatically. Studies on device control via voice are increasingly diverse [2]; Google speech recognition (GSR) [3] is a commonly used system for identifying voice commands. For example, Aripin and Othman [4] researched a voice control system for indoor devices, such as lamps and fans, to support the elderly and the disabled. They suggested GSR to recognize voice commands that would then be transmitted via Bluetooth to control the devices. Iskender et al. [5] proposed the same method to operate vehicles, and Guo et al. [6] used this method to build a spelling correction model.

In this study, we developed a method of Vietnamese voice command recognition using GSR services, as in the above studies [4, 5]. We chose this automatic speech recognition (ASR) system because it is supported on all mobile devices. This service transforms voice commands to text. However, the rate of correct recognition of Vietnamese voice commands returned by Google is not high; the rate of incorrect results in our test was up to 35.06% (Section 3.2). This may be due to the following reasons. First, Vietnamese is an underresourced language [7], and datasets for Vietnamese in this domain are very rare. Vietnamese is also a tonal language. Some syllables have the same phonetic structure but if the tone is different, they have different meanings; so it is possible that GSR does not have a module to handle and identify Vietnamese tones. Secondly, the main goal of GSR is to search for information, so less common commands are often less accurate [8, 9]. For example, in our test, the voice command case is "bật đồng bộ dữ liệu" ("turn on data synchronization"), but GSR often identifies it as "bất đồng bộ dữ liệu" ("data asynchronous"). When testing the exact text search on Google, the first phrase is not common (only 565 search results), while there are 4830 results for the second phrase. These two voice commands differ only in the first syllables "bật" ("on") and "bất" ("no"), which are only different in the tone (the first has a heavier tone and the second has a rising tone).

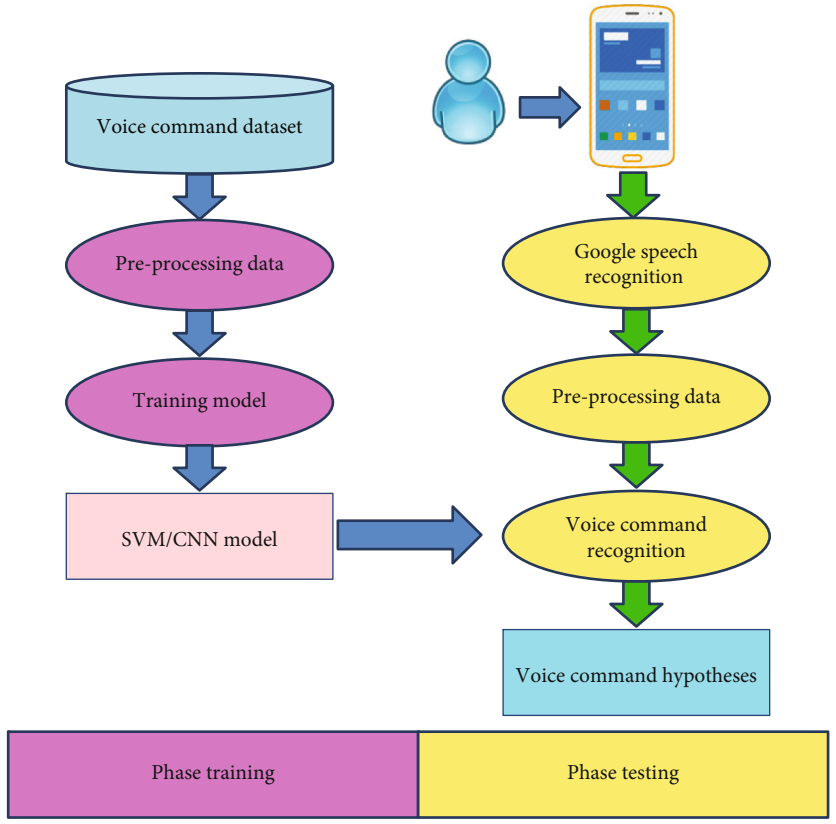

Figure 1: Model of voice command identification system for Vietnamese.

To overcome errors in GSR results, we propose a new method based on a supervised machine-learning approach. The purpose of the proposed method is to transform incorrect results to correct ones. First, we build a voice command dataset that includes the hypotheses of GSR for each corresponding voice command. Based on this dataset, we construct a command classifier, which receives a voice command (returned from GSR) and outputs the corresponding label of the command. Here, the label is in a set of voice commands supported by the system. This method will help the system correctly classify the commands that GSR identified incorrectly.

We took a similar approach to classifying text by category. There are also many text classification approaches based on supervised learning methods. Kumaresan et al. [10] showed the ability to classify spam by naive Bayes, SVM, K-nearest neighbors, and artificial neural network algorithms. The results show SVM as an effective classifier for such problems. Besides, the set of word2vec pretrained vectors yielded very good classification results when using a simple CNN model with only a convolution layer [11, 12]. In this study, we propose a correction system using support vector machine (SVM) and convolutional neural network (CNN) models.

The next part of the paper is organized as follows. Section 2 covers our proposed method. Section 3 details the Vietnamese voice command dataset, the experimental results of the test, and the evaluation. Section 4 describes our application for mobile device control by voice command. Section 5 contains our conclusions and perspectives.

2. Materials and Methods

2.1. Overview Architecture. The method we propose is described in Figure 1. First, for each input voice command, we take five hypotheses returned by the GSR service arranged in order of probability from high to low. Thus, each voice command will be represented by a set of 5 strings (hypothesis 1, hypothesis 2, hypothesis 3, hypothesis 4, and hypothesis 5). In the voice command dataset, each of these hypotheses will correspond to an exact voice command string. The model training process will use this dataset to create a model of correction by using an SVM or CNN model. During the correction process, the machine-learning system uses this model to categorize each voice command as one of the previously described commands. The training of each correction model will be described in the next section.

2.2. Error Correction Model Using SVM. In the data preprocessing step, we first use the bag-of-words (BOW) algorithm [13]. This is a simple model commonly used in natural language processing problems and information gathering. In this model, text is represented as a bag-of-words, ignoring grammar and word order. The BOW model is often used in text classification algorithms in which the frequency of each word appears as a feature for training the

classifier. The result of this step is that each word will be represented by an index showing the position of the word in the word list.

Next, in the dataset, we find that some words appear in many voice commands. For example, in the following two voice commands: "hãy gọi điện thoại" ("please call") and "hãy gửi tin nhắn" ("please send a message"), it is clear that the word "hãy" ("please") appears in both voice commands but does not provide information that helps to distinguish between the commands. The term frequency-inverse document frequency (TF-IDF) algorithm enhances the value of distinctive words and reduces the value of words that appear often (less discriminatory) [14].

The TF-IDF is the weight of a word in the text obtained through a statistics process. This value shows the importance of this word in the text. Term frequency (TF) is used to estimate the frequency of words in the text (Equation (1)). Each text has its own unique length, so the number of occurrences of the word may be different; TF is the ratio of number of occurrences of the word divided by the length of the text (the total number of words in that text).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}. \quad (1)$$

In Equation (1), $n_{i,j}$ is the number of occurrences of the word (t_i) in the text d_j, and the denominator is the total number of words that appear in the text d_j.

Inverse document frequency (IDF) is a quantity used to estimate the importance of a word. In calculating a TF value, the words are considered equally important. However, some words are commonly used but not important to the meaning of the text; for example, linking words or prepositions. We need to reduce the importance of those words using IDF. This value is calculated by

$$idf_j = \log \frac{|D|}{|j : t \in d_j|}, \quad (2)$$

in which $|D|$ is the total number of texts in the sample set D and $|j : t \in d_j|$ is the number of texts containing the word t. However, if the case in set D does not contain any text containing the word t, the above expression will be divided by 0. Therefore, we often add 1 to the denominator as $1 + |j : t \in d_j|$. Then, the TF-IDF value is calculated by

$$tf_{idf}(\mathrm{t}; d; D) = tf(t; d).idf(t; D). \quad (3)$$

Thus, we use the TF-IDF algorithm to represent the value of each word in the BOW. Each voice command will be represented by the following set of values: < index1 >:< value1 >; < index2 >:< value2 >; < index3 >:< value3 >; etc.

The number of attributes (values) of each voice command is the size of that bag, so all voice commands have the same number of attributes. We use the LIBSVM library [15] in this experiment to train the SVM model. SVM is a supervised machine-learning method used for classification and regression problems [16]. A special attribute of SVM is that this method minimizes classification errors and maximizes margins simultaneously. Therefore, SVM is also called the maximization margin classifier. SVM transforms input vectors into a more dimensional space where layers can be separated by a hyperplane. Besides the hyperplane data separation, there are always two parallel hyperplanes built, and the hyperplane dividing the data is that with the largest distance from the two parallel hyperplanes.

Among popular transformation functions (kernel functions), radial basis function (RBF) is used the most. The RBF kernel of two samples (x and x') is calculated by

$$K\left(x, x'\right) = \exp\left(-\gamma \left\| x - x' \right\|^2\right). \quad (4)$$

Therefore, we use the RBF kernel for the SVM model. We need to adjust two parameters: C (with large values of C, all or most of the training points are classified correctly, but we have a smaller margin hyperplane; otherwise, with smaller values of C, that hyperplane misclassifies more points but the margin is larger) and γ (parameter of RBF kernel). We find these parameters using the grid search method with the training data. The optimal parameters are used in the experiment on the test set.

2.3. Error Correction Model Using CNN. The architecture of the CNN network is described in Figure 2. In the preprocessing stage, each word will be represented by the index of words in the BOW. Because the voice commands are of different lengths, the largest command will be chosen as the standard and the smaller commands will be added with special characters <PAD> so that all commands in the dataset are the same size.

2.3.1. Embedding Layer. The first layer we use is the embedding layer, which turns positive integers (indexes of words) into dense vectors of fixed size. This layer is represented by the matrix embedding W of size $[V * E]$, where V is the vocabulary size and E is the vector size (we use the value 128 for E).

2.3.2. Convolution Layer. The convolution layer uses a set of filters (or kernels) to calculate and extract new features from the input matrix. Whether a classification model is effective depends on whether it is able to extract the new features, so this convolution layer is very important in the CNN network. The convolution layer is represented by filter size (usually 3, 4, and 5) and number of filters (usually equal to embedding size, so we use 128 here).

We can imagine the convolution layer acting as a small filter for the input matrix. This filter will multiply the values in the input matrix with the weights in the filter and add them to the specificity of the convolution. After completion, the new matrix is called a convolution feature.

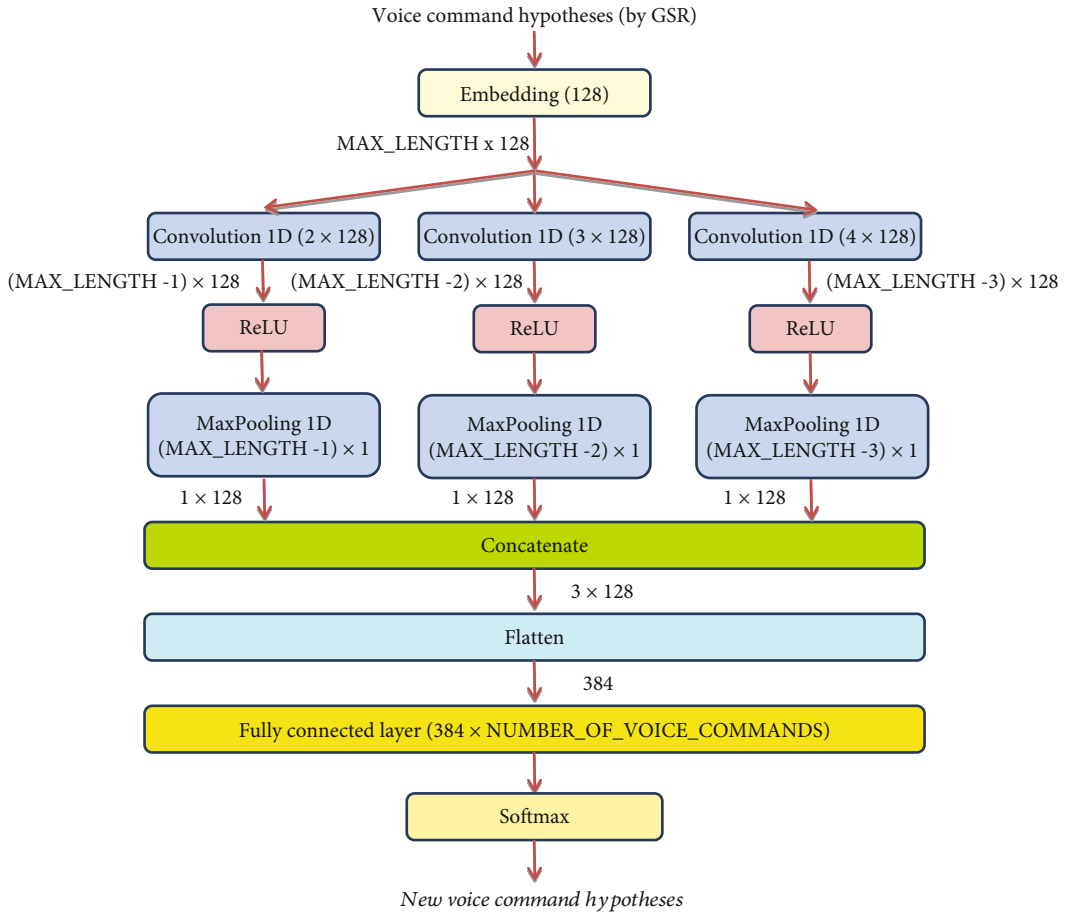

Figure 2: The CNN network architecture used for identifying Vietnamese voice commands.

The value at the point of the coordinates (i, j) in the m^{th} feature map is described in Equation (5):

$$u_{i,m} = \sum_{p=0}^{H-1} \sum_{q=0}^{127} w_{p,q,m} x_{i+p,q} + b_m. \tag{5}$$

(i) $u_{i,m}$ is the i^{th} feature in the m^{th} feature map calculated by 1D convolution layer

(ii) $x_{i+p,q}$ is the q^{th} embedding feature at the $(i+p)^{\text{th}}$ character in voice command

(iii) $w_{p,q,m}$ is the value at the (p, q) position of the m^{th} kernel with m from 1 to M

(iv) b_m is the bias of the m^{th} kernel with m from 1 to M

(v) In our CNN model, the number of feature map M is 128, H is used with values from 2 to 4

2.3.3. Activation Function. After the filters perform convolution on the input matrix, the data will be processed by a nonlinear activation function. If there is no activation function, a neural network layer can only perform a linear transformation on the input data. There are several activation functions, such as sigmoid, tanh, and Rectified Linear Unit (ReLU). In this model, we use the ReLU activation function. The formula of function ReLU is $f(x) = \max(0, x)$ and will replace all negative values to 0; positive values remain. ReLU is used after the convolution layer. Therefore, many models consider the convolution layer and ReLU function one layer.

2.3.4. Pooling Layer. The pooling layer is one of the computational components in the CNN structure. Pooling is the calculation process of the matrix in which the goal after calculating is to reduce the size of the input matrix but highlight its characteristics. There are many pooling operators, such as sum pooling and max pooling, but max pooling is commonly used. In terms of meaning, max pooling will highlight the most characteristics of the input features, so we use max pooling for our CNN model.

2.3.5. Fully Connected Layer. A CNN always has a fully connected layer, and this layer is often designed as the last layer of the network. Neurons in the fully connected layer will fully connect to all neurons in the previous layer. This layer plays the role of classifying data as required by each problem. In this layer, output z_i is calculated by

$$z_i = \sum_{j=1}^{384} w_{i,j} x_j. \tag{6}$$

2.3.6. Softmax Layer. This layer takes as input vector of K real numbers and normalizes it into a probability distribution. The value y_i of this layer is calculated by Equation (7). In this equation, K is the number of voice commands.

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}. \tag{7}$$

We designed the CNN model to return the probability distribution of voice commands in the vocabulary set. So, for other voice commands that are not in the vocabulary set, these probabilities are usually very small, and the system will use this feature to exclude these voice commands. The goal of the system is to correctly identify the voice commands in the list that have been selected previously for each specific application. In the case of adding new voice commands to the system, the following tasks should be done: collecting data of these commands using the GSR service and retraining the model with checking carefully the performance of the system on the validation set.

We trained our CNN model with Adam optimization (learning rate = 10^{-3}). The training was carried out with 200 epochs. The model with the lowest validation loss was used in the test set.

3. Results and Discussion

In this section, we conduct an assessment of the effectiveness of machine-learning applications in improving GSR's voice command recognition results. We built the dataset based on GSR service results, evaluating the rate of incorrect results from GSR and when applying the machine-learning algorithms.

3.1. Building a Dataset. The dataset was collected using the GSR service. For each audio input, GSR will return up to 5 hypotheses arranged in order of high to low probability by the content of the audio input.

To collect data, we made a total of 2350 recordings corresponding to 22 types of voice commands using a Samsung Galaxy A8+ smartphone in a quiet environment (Figure 3). The collected data was synchronized to a real-time database (Firebase) immediately, making it convenient to collect data on multiple smartphones. Voice commands in the dataset are as follows: "Make a call," "Make a message," "Add contacts," "I want to send mail," "Alarm appointment," "Find the directions," "Open the application," "Search," "Weather forecasts," "Volume Up," "Volume Down," "Turn on Wi-Fi," "Turn off Wi-Fi," "Turn on network data," "Turn off network data," "Turn on Bluetooth," "Turn off Bluetooth," "Bright screen," "Dark screen," "Turn on data synchronization," "Turn off data synchronization," and "Open the song." These are the most common commands used to control smart devices. This dataset is named "Vietnamese Voice Commands".

3.2. Evaluating the Accuracy of the GSR System. We use two measurements to measure the accuracy of GSR. The first is called the command error, which allows us to compare the output hypothesis (hypothesis with the highest probability of the returned results) of the GSR with the voice command content. If two strings are different, we consider the command having an accuracy of 0; if the strings are the same, the accuracy is equal to 1. The command error is 1 minus the average accuracy of all the collected statements.

The second measure we use is the word error rate (WER) measurement, which measures the deviation of the output hypothesis (hypothesis with the highest probability of all results returned) with the contents of the voice command. The WER is the minimum number of steps needed to add, replace, or delete a word to convert hypothetical text into reference text [17] (Equation (8)). The WER is calculated by an algorithm based on Levenshtein's distance algorithm.

$$\mathrm{WER} = \frac{1}{N_{\mathrm{ref}}} \sum_{k=1}^{K} \min_r d_L(\mathrm{ref}_{k,r}, \mathrm{hyp}_k). \tag{8}$$

In Equation (8), $d_L(\mathrm{ref}_{k,r}, \mathrm{hyp}_k)$ is the Levenshtein distance between reference text $\mathrm{ref}_{k,r}$ and hypothetical text hy p_k. K, N_{ref} is the size of hypothetical text and corresponding reference text.

When testing GSR on our test set, we obtained a command error of 35.06% and WER of 22.25%. So, the results returned by GSR had a high error rate. In the following section, we describe the experimental results using SVM and CNN machine-learning models to improve the accuracy of the voice command recognition system.

3.3. Performance Evaluation. To evaluate the performance of the voice command recognition system, we recommend the following measures:

(i) P1: incorrect (%) rate voice command, which is calculated as

$$\mathrm{P1} = 100 * \frac{\text{Number of voice commands that model recognized incorrectly}}{\text{Number of voice commands in test set}} \tag{9}$$

(ii) P2: incorrect rate (%) WER in voice command (Equation (8))

(iii) R1: number of commands that Google recognized incorrectly and our model recognized correctly

(iv) R2: number of commands that Google recognized correctly and our model recognized incorrectly

(v) R3: number of commands that Google and our model recognized incorrectly

3.4. Experiments

3.4.1. Dataset Augmentation. After the construction process, we obtained the dataset (2350 samples of 22 voice commands, each of which consisting of up to 5 hypotheses by GSR arranged in order of probability from high to low). We used 20% of the dataset for tests. Due to the specificity of the test set, it was used to check the accuracy of the

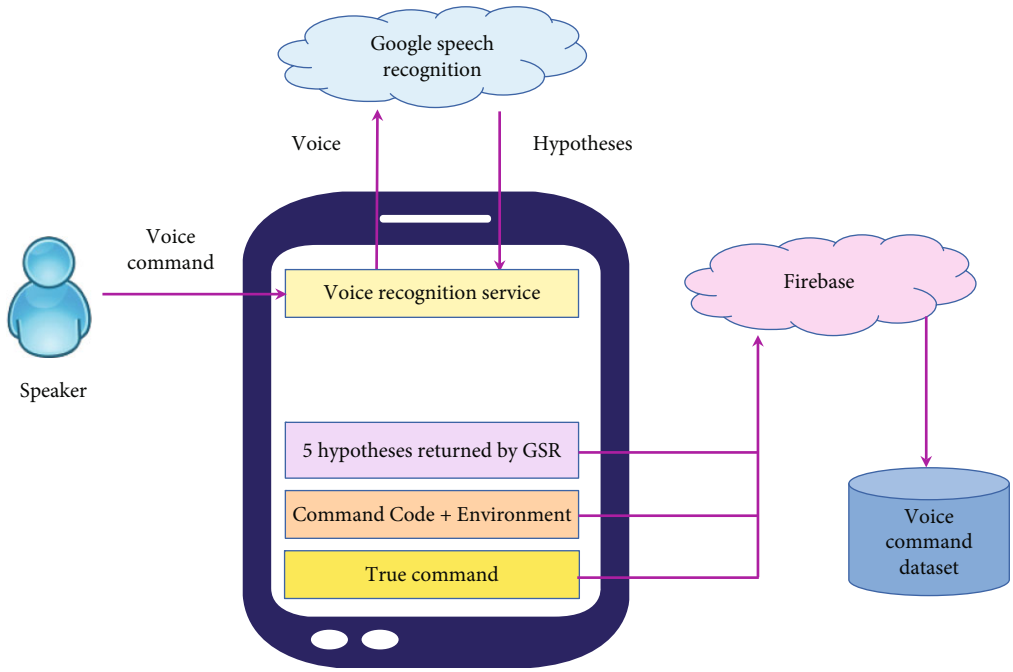

Figure 3: The process of collecting voice command data on an Android mobile device platform using GSR.

machine-learning model, so the test set did not change. For test data, in the 5 hypotheses returned by Google, only one hypothesis of the highest probability for each command was used.

We used 70% of the remaining dataset for training and 10% for the validation set. The training set can be changed to improve the machine-learning model. Therefore, we constructed three types of training datasets as follows:

(i) 1-BEST: use the best hypothesis returned by Google (with the highest probability)

(ii) TYPE-1: combine N-best ($N \geq 2$) hypotheses into one hypothesis. For example, with $N = 2$, if a voice command corresponds with two GSR hypotheses "word_1 word_2" and "word_3 word_4 word_5," then we have a new hypothesis: "word_1 word_2 word_3 word_4 word_5"

(iii) TYPE-2: keep N-best ($N \geq 2$) hypotheses as N-independent hypotheses. For example, with $N = 2$, if a voice command corresponds with two GSR hypotheses, we have two independent hypotheses

Thus, it can be seen that in the TYPE-2 training set, the more N-best hypotheses occur, the more the number of training commands increases, but the command length does not increase. As for the TYPE-1 training set, the length of each command will increase as best hypotheses combine, but the number of commands will not increase.

3.5. Experiment Using the SVM Model. In the voice command identification test using the SVM model, we only used the N-BEST dataset and TYPE-1. The TYPE-2 set had a relatively large amount of data, so it could not be used effectively with the SVM model [18]. The results of the experiment are described in Table 1. Based on the results from Table 1, for the training set 1-BEST, the incorrect ratios P1 and P2 are the lowest (7.08% and 6.46%). For the training set TYPE-1 with $N = 4$ and 5, the error rate (P1) is nearly the same, but the incorrect P2 rate of the $N = 5$ is much lower than the $N = 4$. It can be seen that the denser training set (TYPE-1 will combine N hypotheses into 1 hypothesis, so the length of command is increased when N is increased) has a higher error rate in P1. However, the gap of the incorrect ratios between different training sets is not large.

Table 1: Accuracy of recognition of voice commands using the SVM model.

Training set	P1 (%)	P2 (%)	R1	R2	R3
Train 5-best, TYPE-1	9.01	7.70	96	0	42
Train 4-best, TYPE-1	9.03	8.19	95	0	43
Train 3-best, TYPE-1	9.87	8.34	92	0	46
Train 2-best, TYPE-1	8.58	6.83	98	0	40
Train 1-best	7.08	6.46	105	0	33

3.5.1. Experiment Using the CNN Model. Test results of identification of voice commands using the CNN model are described in Table 2. We noticed that the rate of error for training set TYPE-2 was lower than that of training set TYPE-1. This suggests that with CNN, more data results in lower error rates. For example, when the number of training data increases 5 times (e.g., Train 5-best of the training set

Table 2: Accuracy of recognition of voice commands using the CNN model.

Training set	P1 (%)	P2 (%)	R1	R2	R3
Train 5-best, TYPE-1	9.3	5.93	104	0	34
Train 4-best, TYPE-1	9.66	7.60	93	0	45
Train 3-best, TYPE-1	10.73	8.34	91	3	47
Train 2-best, TYPE-1	9.66	7.05	96	3	42
Train 1-best	7.51	6.43	103	0	35
Train 5-best, TYPE-2	5.15	3.93	115	3	23
Train 4-best, TYPE-2	5.58	3.96	114	0	24
Train 5-best, TYPE-2	6.22	4.61	112	3	26
Train 2-best, TYPE-2	7.08	6.51	108	3	30

TYPE-2 compared to the training set 1-BEST), the error rate decreases by about 2.36% in terms of the P1 measure and by about 2.50% with the P2 measure. The best result was obtained using training set TYPE-2 with $N = 5$: P1 = 5.15% and P2 = 3.93%.

Experimental results show that, with the SVM model, 105 of 138 incorrect identification commands were fixed, and with the CNN model, 115 of 138 incorrect commands were fixed. This demonstrates that the application of machine-learning algorithms has significantly improved GSR's voice recognition capability for voice commands.

4. Discussions

As we described in Introduction, the methods for controlling voice devices rely primarily on an existing speech recognition system due to the infeasible nature of the development and installation of a speech recognition module of high accuracy. Among the speech recognition systems available, recent studies have used the GSR (Google speech recognition) service because of its wide availability on Android devices and the ability to integrate it into device driver programs; examples include a study by Aripin et al. [4] (they built a control system for indoor devices such as lamps and fans to support the elderly and the disabled) and one by Iskender et al. [5] (they used the same method to operate vehicles). However, these studies are not public datasets. Moreover, they only use the direct results of the GSR service. The test results using Vietnamese voice commands (presented in Evaluate the Accuracy of GSR System) showed significant errors (a command error of 35.06% and WER of 22.25%), so we built a module to improve the result based on SVM and CNN models. We also compared the results of these modules and the results of the GSR service. Our proposal method gives the command error rate decreased to 7.08% with the SVM model and 5.15% with the CNN model.

4.1. Application for Mobile Device Control by Vietnamese Voice Command. The architecture of the application is depicted in Figure 4 and consists of three main parts:

(i) Voice recognition service: responsible for recording audio, sending it to the GSR service to handle speech recognition, and receiving textual hypotheses from GSR

(ii) Voice command classification: responsible for assigning command labels to text hypotheses obtained from the voice recognition service. With supervised machine-learning approaches, this section will help correctly recategorize the voice commands that GSR misidentifies. After the classification process is complete, the command label will be sent to the filter and execute function

(iii) Filter and execute functions: responsible for receiving the label of the command from the voice command classifier and performing analysis to filter out the parameters needed to execute the corresponding function. For dual voice commands (including multiple steps), the function filter can be connected directly to the voice recognition unit

This application has been successfully built and deployed on mobile devices running the Android operating system. The results of the experiment with the CNN model were better than those of the SVM model, so in this system, we installed only the CNN model. Table 2 depicts the amount of time needed to identify the voice command (the total time of 10 repetitions for each command) on the Android Galaxy A8+mobile device.

The results listed in Table 3 show that the application works stably for voice command recognition. For commands with parameters (commands 1, 2, 3, 6), the processing time was longer than that for commands without parameters. Processing time for voice command execution analysis is not high and mainly consists of the time spent waiting for results from the GSR service.

5. Conclusions

In this study, we have proposed a novel method to identify Vietnamese voice commands using GSR service results. First, we built a dataset for voice commands based on the GSR service. Then, we applied SVM and CNN models to increase the recognition accuracy of voice commands. The results show that using SVM and CNN machine-learning algorithms significantly improved the ability to recognize voice commands

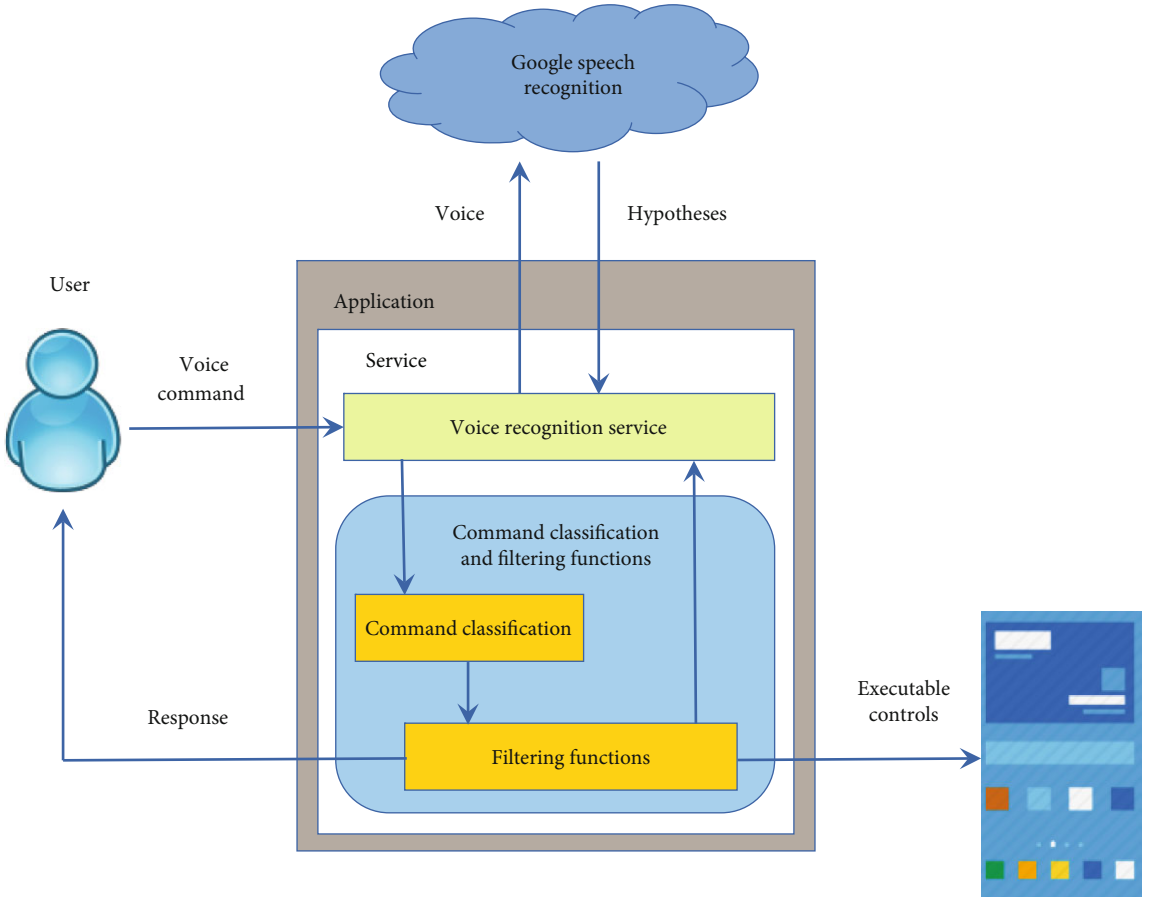

Figure 4: Architecture of application for mobile device control by Vietnamese voice command.

Table 3: Execution time (in milliseconds) to identify smartphone control commands (total times of 10 commands repeated 10 times each). T1 is GSR's time for speech recognition processing. T2 is the processing time for postprocessing voice command analysis by the CNN model.

Command index	T1 (in milliseconds)	T2 (in milliseconds)
1	11934	1676
2	11394	1029
3	13352	216
4	4971	178
5	4639	160
6	10680	217
7	3912	167
8	5795	694
9	3630	182
10	7742	214

compared to results returned directly by GSR. The CNN model gave better identification results than the SVM model; the command error rate decreased from 35.06% with GSR to 7.08% with the SVM model and 5.15% with the CNN model. Indeed, the CNN model promises good results with increasingly diverse training data.

We also offer a method for deploying device control applications via voice command and implementing this application on devices running the Android operating system. Experimental results show that the processing time for analyzing the execution of voice commands compares favourably to the waiting time for results from GSR.

In the future, we plan to conduct training with datasets in many environments and contexts, for example, in outdoor environments and on datasets consisting of various Vietnamese dialects.

Data Availability

The "Vietnamese Voice Commands" dataset used to support the findings of this study are included within the supplementary information files.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Supplementary Materials

Vietnamese voice commands. *(Supplementary Materials)*

References

[1] M. Jeffs, *OK Google, Siri, Alexa, Cortana; can you tell me some stats on voice search?*, 2019, https://edit.co.uk/blog/google-voice-search-stats-growth-trends.

[2] J. W. Jenness, L. N. Boyle, J. D. Lee et al., "In-vehicle voice control interface performance evaluation," *Final report, Report No. DOT HS 812 314*, National Highway Traffic Safety Administration, Washington, DC, 2016.

[3] X. Lei, A. W. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Interspeech 2013*, pp. 662–665, ISCA, Lyon, France, 2013.

[4] N. Bt Aripin and M. B. Othman, "Voice control of home appliances using Android," in *2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS)*, pp. 142–146, Malang, Indonesia, 2014.

[5] A. Iskender, H. Ucgun, U. Yuzgec, and M. Kesler, "Voice command controlled mobile vehicle application," in *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, Turkey, 2017.

[6] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5651–5655, Brighton, UK, 2019.

[7] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: a survey," *Speech Communication*, vol. 56, pp. 85–100, 2014.

[8] C. Chelba, J. Schalkwyk, B. Harb et al., *Language modeling for automatic speech recognition meets the web: GoogleSearch by voice*, University of Toronto, 2012.

[9] M. Schuster, "Speech recognition for mobile devices at Google," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 8–10, Springer, Berlin, Heidelberg, 2010.

[10] T. Kumaresan, S. SanjuShree, K. Suhasini, and C. Palanisamy, "Machine learning algorithms used in spam filtering," *International Journal of Advanced Information and Communication Technology*, vol. 1, no. 7, 2014.

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, pp. 3111–3119, The MIT Press, 2013.

[13] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[14] J. E. Ramos, "Using TF-IDF to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning (iCML-2003)*, vol. 242, pp. 133–142, Piscataway, NJ USA, 2003.

[15] C.-C. Chang and C.-J. Lin, "LIBSVM," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[16] V. Vapnik, "The Nature of Statistical Learning Theory," in *Second Workshop On Statistical Machine Translation (ACL 2007)*, Springer-Verlag, 1995.

[17] M. Popović and H. Ney, "Word error rates: Decomposition over POS classes and applications for error analysis," in *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pp. 48–55, Prague, Czech Republic, 2007.

[18] A. K. Menon, *Large-scale support vector machines: algorithms and theory*, Research Exam, University of California, San Diego, 2009.