An Attention-Based Command Detection Model to Allow Natural Language in Voice Control System

Jie HE, Ze-xing LI and Bin-tang YANG*

The State Key Laboratory of Mechanical System and Vibration, Shanghai JiaoTong University, Shanghai 200240, China

*Corresponding author

Keywords: Man-machine Interaction, Voice control system, Sequence-to-sequence modeling.

Abstract. Interaction with machines using speech has been a popular research topic in recent years. Considerable amount of work focuses on the improvement of voice control systems which utilize automatic speech recognition (ASR) and natural language processing (NLP) technologies to recognize user’s speech and extract executable commands. The existing voice control systems usually do not take into count the diversity and richness of natural language and require users to follow pre-defined keywords or grammar rules. To address this limitation, we designed and implemented a voice control system that supports natural language, utilizing an attention-based command detection model. Our system supports flexible voice instructions and the user does not need to follow any pre-defined rules. An Arduino 4WD robot car was also built in this paper to verify the system. In the experiments, the accuracy of command detection on natural language reaches 0.993 in our developed dataset. Besides, the realization of the 3-DOF (degree of freedom) motion control on our robot car suggests the feasibility of using our proposed system to control any functionality or behavior of the hardware system. Our work improves the flexibility and usability of voice control systems by applying technologies in the domain of NLP.

Introduction

Service robots can be applied to people’s life in a wide range of domains, such as health care, traffic control, public services, industrial automation and so on. Human-robot interaction (HRI) is the most important and challenging area in robotics system, because robot is designed to assist people with the bottom line of being able to be controlled by humans. It is obvious that interacting with speech is more natural and convenient than using hand gestures, eye motion, etc. With the development of automatic speech recognition (ASR) and Natural Language Processing (NLP) technologies, researchers gain the opportunity to design and implement speech based HRI [1-3].

Several works relating to speech based HRI is described here. Brandi House et al. developed a voice controlled robot arm named VoiceBot where the non-verbal voice specifies continuous control signals [1]. Their work can be seen as the beginning research in the field of voice control systems. However, VoiceBot does not recognize natural spoken language and only supports a set of pre-defined voice command to control the arm, where the user has to be well trained before using the system properly. In [2], a client-server based architecture for controlling multiple robots simultaneously through predefined voice commands is developed. While the application of speech recognition allows to interacting with spoken language, this architecture only supports seven pre-defined commands to control the robots (directions and speed) and the voice command should follow the fixed grammars. Recently, similar voice control system is developed in [4,5], where the object is controlled via a set of pre-defined voice commands. The limitation of existing voice control systems is that users’ voice need to follow the pre-defined commands or grammars in order that the system can retrieve command from recognized voice. For instance, when a user says “Please go”, the robot in [2] would not response for its failure to match “Please go” with any pre-set command, while the correct command “forward” can be easily implied by a human.

In Artificial Intelligence (AI), neural network models are dedicated to imitating the way human processes information, such as translating text from one language to another or detecting a cat in a
picture. In recent years, a neural network structure, the encoder-decoder with attention mechanism, shows high capacity on sequence-to-sequence tasks [6,7]. The extraction of command in voice control system can be also seen as a sequence-to-sequence task, which turns a sentence into a command.

In this paper, we developed a novel voice control system which applied the attention-based encoder-decoder structure to support natural language. Our system allows users to freely express control intentions in natural language instead of following pre-defined rules. Besides, multiple commands and digital parameters are also supported in our system. The rest of this paper is organized as follows: the proposed voice control system is illustrated in section 2. Experiment is conducted in Section 3. Finally we conclude our work in Section 4.

Proposed Voice Control System based on Command Detection Model

Description of Voice Control System

Our proposed voice control system is mainly based on the advanced automatic speech recognition (ASR) and natural language processing (NLP) technologies, which allow extracting executable command from recognized oral voices. As shown in Fig. 1, our voice control system mainly consists of a mobile application, a server and robot. User’s voice is firstly passed to the mobile application, where the built-in ASR module developed by IFlyTek recognizes the voice. The recognized sentence is then sent to the command detection module in the server, which extracts executable command from the sentence. Finally, the robot executes the command received from the server. The ACKs in Fig. 1 represent the corresponding execution status or results in the system.

![Figure 1. Architecture of the proposed voice control system.](image)

Compared to the voice control system proposed in [2], the proposed system in this work applies a command detection model to parse recognized voice in the form of text. Our system does not require users’ voice to follow the pre-defined rules, such as particular grammars and keywords. The details of our model will be described in the following section. Besides, our system implements one single server to perform both network communication and command extraction tasks, instead of two servers in [2]. In addition, less network communications is involved in our system with lower Internet latency and fewer connection failures.

The performance of the voice control system mainly depends on the efficiency of speech recognition and the accuracy of command extraction. The third party IFlyTek is chosen as our ASR service provider, which features high recognition accuracy and provides user-friendly APIs. Therefore, in the basis of IFlyTek’s SDK for speech recognition, this paper mainly focuses on the command extraction part. Specifically, we designed a command detection model to perform comment extraction from recognized oral voices, which consists of an attention-based encoder-decoder structure. The efficiency of the attention mechanism and the encoder-decoder structure enables intelligently extracting command from the recognized speech.
Attention-Based Command Detection Model

The Encoder-Decoder Structure. Since both the user’s speech (in text form) and the desired command can be seen as a sequence of words, our command detection task can be regarded as a sequence-to-sequence problem. We intuitively applied the encoder-decoder structure developed in the field of Natural Language Processing (NLP). An encoder-decoder structure, which consists of an encoder and a decoder, is a popular neural network structure for its high capacity to address sequence-to-sequence problems. Its wide applications include neural machine translation (NMT) [7], speech recognition [8], and so on. These tasks have one thing in common hat they have a source sequence and a target sequence. For instance, in neural machine translation, a source sequence can be a sentence from one language, and the corresponding target sequence is the translation of the source sentence in another language.

The basic idea of encoder-decoder structure can be explained as modeling the conditional probability \( p(y|x) \) of translating a source sequence \( x = (x_1, x_2, \ldots, x_n) \) to a target sequence \( y = (y_1, y_2, \ldots, y_m) \). The encoder takes as input the source sequence \( x = (x_1, x_2, \ldots, x_n) \), where \( x \) is usually represented by a vector, and generates a representation \( S \) for \( x \). The decoder takes \( S \) and produces one target token \( y_j \) at a time. Hence, \( p(y|x) \) can be detailed as,

\[
\log p(y|x) = \sum_{j=1}^{m} \log p(y_j|y_{1,2,\ldots,j-1}, S),
\]

(1)

Generally, the encoder uses recurrent neural networks (RNNs) [9], since RNNs have been proven effective on associating sequential information from input sequence. In some cases, convolutional neural networks (CNNs) are also applied in the encoder. The decoder employs RNNs for its recurrent nature so that the probability distribution over the candidate tokens in each step can be calculated as,

\[
p(y_j|y_{1,2,\ldots,j-1}, S) = f(s_j),
\]

(2)

where \( f \) is the transformation function which ends with a softmax operation to produce a vector of probabilities \( p = (p_1, p_2, \ldots, p_m) \), and \( s_j \) is the hidden state vector of \( j_{th} \) step during decoding. The final predicted token is decided by the index of the maximal value in \( p \).

To make the whole encoder–decoder model work, a training dataset is required to fit the model where each data sample consists of a source sequence and a target sequence. We train the model to minimize the loss function \( J(\theta) \):

\[
J(\theta) = \sum_{(x,y) \in D} \log p(y|x; \theta),
\]

(3)

where \( D \) is the training dataset, and \( \theta \) is the parameters in the model. \( J(\theta) \) is minimized by repeatedly updating \( \theta \) through stochastic gradient descent (SGD) technique.

Attention Mechanism. The original encoder-decoder structure proposed in [9] cannot effectively make use of the information in the input sequence since its encoder outputs one single vector with fixed length. Later research has adopted the attention mechanism on the decoder [9] to associate more information from the input sequence [6]. The attention mechanism was firstly introduced on Image Processing [10] which imitated human’s attention behavior to focus on a part of a picture instead of on the whole one. The raw attention mechanism can be described as calculating a weighted summation vector (attention vector) \( C \):

\[
C = \sum_{i=1}^{n} \alpha_i \cdot v_i,
\]

(4)

where \( v_i = (v_{i,1}, \ldots, v_{i,s}) \) is value vector, and \( \alpha_i = (\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,s}) \) represents the weights on \( v_i \). In detail, \( \alpha_i \) is a set of normalized attention scores normalized by the softmax function, denoted as:

\[
\alpha_i = \frac{\exp(\text{att}(q, c_i))}{\sum_{j=1}^{n} \exp(\text{att}(q, c_j))},
\]

(5)
where \( att \) is the attention function. The attention function is usually based on the dot product operation or Deep Neural Networks (DNNs). In practice, the attention function is often applied to the context vectors themselves, i.e., \( v_t = c_t \).

We now detail how a recognized speech sequence is converted to its target command via the attention based encoder-decoder structure.

**The Encoder Model.** Inspired by the works in [6,7], our encoder also uses one of the RNN’s variants, the Long-short Term Memory (LSTM) [11,12], which has an improved memory mechanism to associate information within long input sequence. An Embedding layer is also applied to convert indexes into vectors. To format the input and output as sequences, both of the input sentence and the output command are divided at word level excluding the digits (in string format) which are divided at character level. Besides, a \(<\text{EOS}>\) token was appended to both input and output sequences to mark the sequence end. We use two fixed dictionaries, which are initially built upon all the train samples, to convert tokens into their corresponding indexes in the dictionaries and vice versa for the input and the output sequences respectively. Hence, the input sentence (recognized speech), can be represented by a list of indexes \( I = (i_1, i_2, \ldots, i_s) \), where \( i_t \) is the index of the \( t \)th token in the input dictionary and \( s \) is the sequence length.

The Embedding layer takes the input sequence \( I \) and converts it into a sequence of vectors:

\[
(x_1, x_2, \ldots, x_s) = \text{Emb}(i_1, i_2, \ldots, i_s),
\]

(6)

where \( \text{Emb} \) is the Embedding layer which simply looks up the vectors for given indexes. The LSTM reads the output of the Embedding layer \( X = (x_1, x_2, \ldots, x_s) \), and iterates \( x_t \) in \( X \) with the calculation,

\[
h_t = C_{LSTM}(x_t, h_{t-1}),
\]

(7)

where \( h_t \) is the hidden state vector for each iteration step in the input sequence, and \( C_{LSTM} \) is the LSTM cell that performs calculations based on input vector, hidden state and cell state vectors, which results in updated hidden state and cell state vectors [12].

To incorporate with the attention mechanism, we keep all the hidden vectors \( H_s = (h_1, h_2, \ldots, h_s) \) during the encoding stage. Besides, the single LSTM is replaced by a bi-directional LSTM which performs the Eq. 7 in both forward and backward directions, and generates the forward hidden vector \( h_t^f \) and backward hidden vector \( h_t^b \) respectively. Finally, we get \( \tilde{H}_s = (\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_s) \) as the representation of the input sentence, where \( \tilde{h}_i \) is the concatenation of \( \tilde{h}_i^f \) and \( \tilde{h}_i^b \).

**The Decoder Model with Attention Mechanism.** Our decoder also employed the LSTM cell to recurrently generate output tokens and form a command in the end. At each step, the model predicts a new token \( y_j \), given the context vector \( \tilde{H}_s \), previous hidden state \( s_{j-1} \) and the previous predicted token \( y_{j-1} \). The \( s_j \) in Eq. 2 is computed by:

\[
s_j = C_{LSTM}(s_{j-1}, y_{j-1}, C_j),
\]

(9)

where \( C_j \) is the attention vector calculated by Eq. 4 and Eq. 5. We use a fully connected layer with the \( \text{tanh} \) activation function as the \( att \). To initiate the decoder, the last hidden state vector \( \tilde{h}_s \) from the encoder is used as the initial hidden state \( s_0 \), and a \(<\text{SOS}>\) (start-of-sequence) token is used as the initial token \( y_0 \).

**Experiments**

**System Configuration**

To test our voice control system, we built an Arduino 4WD robot car which mainly consists of the Arduino Mega 2560 as the main control board, the L298N as the servo driver board, and the
NodeMCU (ESP8266) as the network communication module. The MPU9250 nine-axis gyroscope is used for attitude feedback. Fig. 3 is the picture of our actual robot car.

![Figure 3. Picture of the robot car.](image)

When the NodeMCU receives command from our voice control system through the network, it converts the command into callable function names and parameters (if necessary) and passes them to Arduino through UART protocol. Arduino receives the function names and parameters, and calls these functions (with parameters) one by one to drive the motor. Note that the all the possible function names parsed from the command have already been implemented in the Arduino program. The execution status and result will be sent back to the voice control system.

To receive user’s voice and convert it into sentence (text form), we have developed an Android application with API version 23. A cloudy server with public IP has been established to connect the mobile application and the robot through network, as well as to host the command detection model. Network communications within the whole system are implemented via the Socket module based on TCP/IP protocol.

**Dataset Preparation and Model Training**

To train our command detection model, we developed a dataset of 105,000 samples where each sample is a pair of a sentence and a target command string, as the sample shown in Table 1. We take into consideration the diversity and richness of natural language and extend to all possible words, phrases and sentence structures for one certain user command or instruction. The data sample in Table 1 can have multiple alternative expressions, as shown in Table 2. It should be noted that, those alternative expressions should not involve any ambiguity and they should be interpreted as the same meaning. Besides, we further extend the dataset by simply replacing the numbers involved in the pairs with random numbers of range [0, 100.00].

To build and train the command detection model, we implemented our code in Python using the platform Keras. Keras is a high-level neural networks API which is written in Python and supports several popular backends, i.e., TensorFlow, CNTK, and Theano. Our LSTM has 256 hidden nodes and the embedding dimension is 128. Based on our dataset, the sizes of our input and output dictionary are 99 and 33 respectively. We trained our model for 20 epochs using Adam [13] as SDG optimization function.

| Source sentence | Please move forward 5.2 meters and then turn left by 30 degrees. |
|-----------------|------------------------------------------------------------------|
| Target command  | move 2.5 1; rotate 30 1                                         |

Table 1. One sample in our train data. Here, the target command consists of two sub-commands separated by the comma, and the second number “1” denotes the unit of the precedent number.
Table 2. Various possible oral expressions for one target command.

| Source sentence 1 | Please go 5.2 meters, then rotate by 30 degrees. |
|-------------------|---------------------------------------------------|
| Source sentence 2 | Can you move 5.2 meters and turn left by 30 degrees? |
| Source sentence 3 | Hello, please move 52 centimeters, then rotate by 30 degrees. |
| Target command    | move 2.5 \( \mathbf{1} \); rotate 30 \( \mathbf{1} \) |

Experiments and Analysis

Theoretically, our proposed voice control system can control arbitrary available functionality or behavior of the hardware incorporating with the hardware’s driver, such as positioning control of a 6-DOF precision drive platform or the procedure control of a CNC machine. In order to verify this, we initially tested the system on our 3-DOF robot car. The experiment is divided into two parts:

**Performance of Command Detection Model.** In order to directly obtain the performance of our command detection model, we exclude the impact of speech recognition by using a set of artificially synthesized sentences instead of using voice input from users. The test results are shown in Table 3, where ENCDEC model is our proposed model. We calculate accuracy simply by counting how many extracted commands are identical to the target commands in the test data. The high accuracy of 0.982 indicates that command detection model has gained capacity to extract commands from sentences. A higher accuracy can be expected if we continue to tune the model parameters during training.

| Model       | Train Data | Test Data | Accuracy |
|-------------|------------|-----------|----------|
| ENCDEC      | 105,000    | 559       | 98.2%    |
| ENCDEC-DP   | 105,000    | 559       | 99.3%    |

ENCDEC-DP is a further improvement on our model which applies the digit placeholder. In practice, the digits involved in the user’s speech are usually recognized directly by our ASR engine. In this case, the model does not need to translate the explicit digits since they are accurately mapped between the source sentence and the target command. In detail, all the digits in input sentence are replaced with enumerated tokens \( < \text{num}_1 >, < \text{num}_2 >, \ldots, < \text{num}_k > \), where \( k \) indicates the \( k \)th digit present in the source sentence. The digits in output sequence which has the same value as the digits in source sentence are also replaced with a corresponding token \( < \text{num}_k > \). When the model infers new sentence, the digits are handled in the same way for input and the enumerated placeholders are reverted back to their true values. The application of digit placeholder ensures that the model can precisely translate the explicit digits. Due to this improvement, the accuracy goes up to 0.993, which corresponds to our observation that most of the wrong outputs from the test on ENCDEC model are related to inconsistency on digits translation. More importantly, using digit placeholder can reduce both encoding and decoding steps which improves the model’s potential to deal with long sentence, since the memory capacity of LSTM decreases with sequence length.

**System-level Experiment with Robot Car.** To test the global performance of the voice control system on our robot car, we integrated the mobile application, cloud service program, command detection module and the robot car driver. Fig. 4 demonstrates how a user’s voice is passed through the system and reaches the robot car as a command.
We conducted 50 tests by giving natural language instructions and measured the car’s motion. The results show that the response accuracy of the whole system reaches 92% which is lower than that of the particular command detection model. This is due to rare failures during the speech recognition, resulting in meaningless text. In addition, we performed system-level verification on the support of natural language. We chose a fixed command “move 2.5 1; rotate 30 1” and synthesized several different expressions (without ambiguity), as shown in Table 4. The results show that the command detection model can translate various oral expressions (of the same meaning) into the same command. Besides, the realization of 3-DOF motion control on the robot car implies the feasibility of voice control on any available functionality or behavior of a hardware system.

It should be noted that 1) there is a certain error between the actual motion of the robot car and the target command. It’s because that the driver and feedback control on the robot car does not have high precision, which is not a key point of our work described in this paper. 2) For all the tested sentences in the above experiments, the control intentions are within the range of the functionality of the control object. Utterances that cannot be performed by the robot car, such as “jump forward”, are not considered in the test either.

| Utterance                                      | Measured Motion               |
|-----------------------------------------------|-------------------------------|
| Please go 2.5 meters, then rotate by 30 degrees.| Moved forward by 2.6 meters and turned left by 35 degrees. |
| Can you move 2.5 meters and turn left by 30 degrees? | Moved forward by 2.5 meters and turned left by 30 degrees. |
| Hello, please move 2.5 meters, then rotate by 30 degrees. | Moved forward by 2.6 meters and turned left by 32 degrees. |

**Conclusion**

In this paper, we proposed a novel voice control system which supports natural language. We extract the command from the recognized speech via a command detection model which leverages the attention-based encoder-decoder model in the domain of NLP. Our experiments show promising results that the proposed voice control system can tolerate user’s various oral expressions and generates desirable commands with the accuracy of 0.993 on our dataset. In addition, the system-level experiment on our 3-DOF robot suggests the feasibility of voice control on arbitrary available functionality or behavior of a hardware system.

Future work will pay attention to the tuning of the encoder-decoder model. The encoder can try some other neural networks such as CNNs or the combination of CNNs and RNNs. In addition, the attention mechanism can be substitute with its variants. We also plan to employ our proposed
system on some precision positioning platforms as well as CNC machines in industry to improve their usability for a wide range of people who have no or little professional operation skills. Besides, voiceprint recognition may be considered be implemented in the mobile application to ensure the operation authentication.

Acknowledgements

The authors gratefully acknowledge the support from the National Key Research and Development Program of China (2017YFF0108000) and the National Natural Science Foundation of China (51775349).

References

[1] B. House, J. Malkin, and J. Bilmes, “The voicebot: A voice controlled robot arm,” in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI ’09. ACM, 2009, pp. 183–192.

[2] K. Shrivastava, N. Singhal, P. K. Das, and S. B. Nair, “A speech recognition client-server model for control of multiple robots,” in Proceedings of Conference on Advances In Robotics, ser. AIR ’13, 2013, pp. 69:1–69:6.

[3] Y. Wang and P. Dong, “The design and implementation of the voice control system of smart home based on ios,” in 2016 IEEE International Conference on Mechatronics and Automation, Aug 2016, pp. 133–138.

[4] A. H. M. Saod, N. A. Harron, F. Ahmad, N. H. Ishak, S. A. Ramlan, A. N. A. Rashid, and M. K. N. Azmi, “Speech-controlled vehicle for manufacturing operation,” in 2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Nov 2016, pp. 490–493.

[5] T. Babu Pulikottil, M. Caimmi, M. Grazia Dangelo, E. Biffi, S. Pellegrinelli, and L. Molinari Tosatti, “A voice control system for assistive robotic arms: Preliminary usability tests on patients,” pp. 167–172, 08 2018.

[6] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” Computer Science, 2014.

[7] M. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” CoRR, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[8] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” Computer Science, vol. 10, no. 4, pp. 429–439, 2015.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” vol. 4, pp. 3104–3112, 2014.

[10] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” CoRR, vol. abs/1406.6247, 2014. [Online]. Available: http://arxiv.org/abs/1406.6247

[11] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” vol. 9, pp. 1735–80, 12 1997.

[12] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with Istm,” Neural Computation, vol. 12, pp. 2451–2471, 1999.

[13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” Computer Science, 2014.