Voice command module for Smart Home Automation

LUKA KRALJEVIĆ, MLADEN RUSSO, MAJA STELLA
Laboratory for Smart Environment Technologies,
University of Split, FESB
Ruđera Boškovića 32, 21000, Split
CROATIA
lkraljev@fesb.hr

Abstract: - Voice control is the most prominent feature of smart home environment. In this paper, we proposed a voice command module that enables users hands-free interaction with the smart home environment. We presented three components required for simple and efficient control of the smart home devices. Wake up word component allows actual voice command processing. Speech recognition component maps spoken voice commands to text and Voice Control Interface parse that text into appropriate JSON format for home automation. We evaluate the possibility of using the voice control module in a smart home environment by separately analyzing each component of the module.

Key-Words: - Smart Home, Speech Recognition, Voice control, Wake-up Word, Commands Parsing

1 Introduction

Smart home environment represents an automation system in which different sensors and intelligent devices work together providing efficient service functions to improve comfortable quality of house living. The smart environment can be seen as a promising way of supporting independent living providing in-home assistance which is especially significant to people with some disabilities [1]. Speech recognition is considered to be a bridge for better and more natural human-computer interaction [2]. Recent advances in automatic speech recognition (ASR) made this technology one of the essential features of the smart home automation (HA) system [3][4]. Our goal is to enable voice control within the smart home laboratory that can be used to control a large number of devices such as lights, plugs, sensors, dimmers, etc. In this paper, we present the architecture of the voice control module which has a role of converting the spoken commands to text and then mapping them to the appropriate actions in the HA system. Module continuously listens and process the sounds from the environment to ensures hands-free interaction. To minimize the number of cases in which the action in the HA system is triggered without actual user intent, we introduced wake up word (WUW) detection mechanism.

2 System Architecture

In this section, we described the architecture of the proposed voice control module. The proposed module consists of Wake-Up Word detection module, Speech recognition engine and Voice command interface as shown in Fig 1.

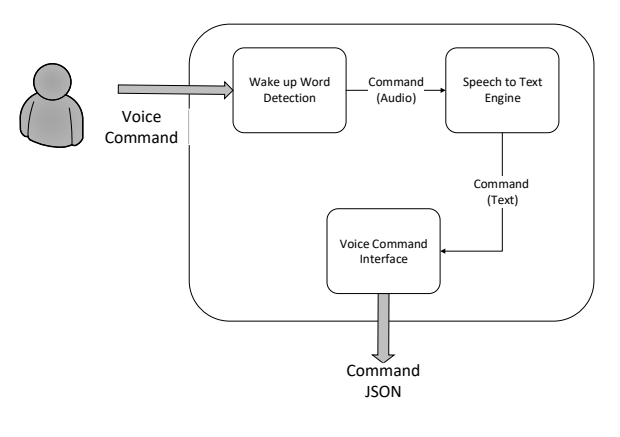

Figure 1 Voice Command Module

First, we will give a brief introduction inside deep network architecture behind our WUW and ASR components followed by the description of each component.

2.1 Deep Network Architecture

Because of the ability to handle sequential information almost all end-to-end speech recognition systems use some recurrent neural network in their pipeline [5]-[8]. In the RNN, the internal representation of dynamic speech features is formed by feeding the low-level acoustic features into the hidden layer together with the recurrent

hidden features from the history. As an improvement of conventional RNN, Bidirectional Recurrent Neural Networks (BRNNs) can make use of future context. Data is processed both directions with two separate hidden layers by splitting the state neurons of a regular RNN in a part that is responsible for the positive time direction and part for the negative time direction which are then fed forwards to the same output layer as shown in Fig2.

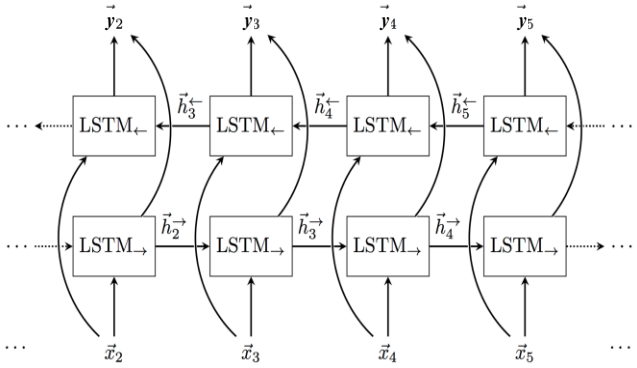

Figure 2 Bidirectional LSTM

Outputs from forward states are not connected to inputs of backward states, and vice versa [9]. Given the input sequence $x = (x_1 \ldots x_t)$ BRNN computes the forward hidden sequence $\vec{\mathbf{h}}$, the backward hidden sequence $\overleftarrow{\mathbf{h}}$ and output sequence $y=(y_1 \ldots y_t)$ with the following iterative process:

$$\overrightarrow{\mathbf{h}_t} = f\left(\mathbf{W}_{x\vec{h}}\mathbf{x}_t + \mathbf{W}_{\vec{h}\vec{h}}\vec{\boldsymbol{h}}_{t-1} + \mathbf{b}_{\vec{h}}\right) \quad (1)$$

$$\overleftarrow{\mathbf{h}_t} = f\left(\mathbf{W}_{x\overleftarrow{h}}\mathbf{x}_t + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\boldsymbol{h}}_{t+1} + \mathbf{b}_{\overleftarrow{h}}\right) \quad (2)$$

$$\mathbf{y}_t = \mathbf{W}_{\vec{h}y}\overrightarrow{\mathbf{h}_t} + \mathbf{W}_{\overleftarrow{h}y}\overleftarrow{\mathbf{h}_t} + \mathbf{b}_y \quad (3)$$

Learning RNN parameters $\mathbf{W}_{hy}, \mathbf{W}_{xh}$ and $\mathbf{W}_{hh}$ goes by using backpropagation through time (BPTT) algorithm. For learning those weight matrices of an RNN unfolds the network in time and propagates error signals backwards through time. BPTT can be viewed as extension of the classic backpropagation algorithm for feed-forward networks, where the stacked hidden layers for the same training frame t, are replaced by the T same single hidden layers across time *t = 1,2,3,...,T.* The use of the state space in the RNN enables its representation and learning of sequentially extended dependencies over arbitrarily long sequences but in practice has been shown that they are not capable of looking far back into the past in many types of input sequences. Generally, it is difficult to train RNN with commonly used activation functions (*tanh*, *sigmoid*, *RELU*) due to the exploding and vanishing gradient [10]. One of the solutions is to embed the memory structure like long-short-term memory cells (LSTM) [11]. The structure of a single LSTM memory cell is illustrated in Fig 3.

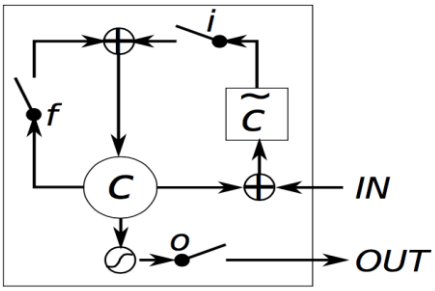

Figure 3 Architecture of simple LSTM

For LSTM, the computation at the time step t can be formally written as follows:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \bullet \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \bullet tanh\left(\boldsymbol{W}^{(xc)}\boldsymbol{x}_t + \boldsymbol{W}^{(hc)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(c)}\right) \quad (4)$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^{(xo)}\boldsymbol{x}_t + \boldsymbol{W}^{(ho)}\boldsymbol{h}_{t-1} + \boldsymbol{W}^{(co)}\boldsymbol{c}_t + \boldsymbol{b}^{(o)}\right) \quad (5)$$

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(xi)}\boldsymbol{x}_t + \boldsymbol{W}^{(hi)}\boldsymbol{h}_{t-1} + \boldsymbol{W}^{(ci)}\boldsymbol{c}_{t-1} + \boldsymbol{b}^{(i)}\right) \quad (6)$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(xf)}\boldsymbol{x}_t + \boldsymbol{W}^{(hf)}\boldsymbol{h}_{t-1} + \boldsymbol{W}^{(cf)}\boldsymbol{c}_{t-1} + \boldsymbol{b}^{(f)}\right) \quad (7)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \bullet tanh\left(\boldsymbol{c}_t\right) \quad (8)$$

Above formulas describe five different types of information at time t representing the input gate, forget gate, cell activation, output gate, and hidden layer, where output σ is the logistic *sigmoid* function, $\boldsymbol{W}$ is weight matrices connecting different gates and $\boldsymbol{b}$ are the corresponding bias vectors.

2.2 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [12] is one of the most successful sequence-to-sequence learning algorithms for RNNs. CTC is an objective function that allows an RNN to be trained for sequence transcription tasks without requiring any prior alignment between the input and target sequences. It uses a *softmax* output layer to define a separate output distribution P(k|t) at every step t over all possible label sequences, conditioned on a given input sequence. The output layer contains a single unit for each of the transcription labels (characters, phonemes, musical notes, etc.), plus an extra unit referred to as the 'blank' which corresponds to a null. Alignment of CTC π is defined by length T sequence of blank and label

indices. The probability P(π|x) is a product of the emission probabilities at every time step.

$$P(\pi|x) = \prod_{t=1}^{T} P(\pi_t|t,x) \qquad (9)$$

For a given transcription sequence, there are as many possible alignments as there is different ways of separating the labels with blanks. Given this distribution, an objective function can be derived that directly maximizes the probabilities of the correct labeling. Since the objective function is differentiable, the network can then be trained with BPTT algorithm, where we try to minimize the CTC objective function:

$$CTC(\boldsymbol{x}) = -\log P(\boldsymbol{y}^*|\boldsymbol{x}) \qquad (10)$$

where $\boldsymbol{y}^*$ is target transcription.

2.4 Wake-up Word Detection

Voice command module continuously receives and process sound. The goal of Wake-up Word (WUW) detection module is reducing the additional number of cases in which ASR preforms Speech to Text (SST) operations at the same time minimizing unwanted triggers of Voice Command Interface (VCI). Therefore, WUW spotting module can be viewed as a continuous identification process of the predefined wake-up word while rejecting all other words, sounds, and noises.

In this work, we presented a context-aware keyword spotter which consists of unidirectional RNN (front-end) and decoder (back-end) similar to the one in [13]. RNN is trained as end-to-end ASR with CTC to transcribe the input speech to a sequence of character labels. Given the purpose, keyword spotting compared to large vocabulary ASR requires simpler architecture suitable for real-time low-power systems; hereof network consists of two unidirectional LSTM layers and a *softmax* output layer. At each frame, the soft output of the RNN is fed into back-end decoder for computing posterior probabilities of keywords. Decoder takes labels probabilities (characters, word-boundary and CTC blank) and converts them to character level output sequences.

Word-boundary label is responsible for detecting keyword as a substring of other words, i.e., decoders network is bounded by the word-boundary label "_". At each time frame decoder updates state by multiplying networks input probabilities with RNN labels probabilities. Nodes of the network contain either character label or word-boundary label followed by CTC blank. Decoder network probability is calculated as a sum of all incoming probabilities, and when negative log posterior probability is below a threshold, then the keyword is detected where the threshold is proportional to the number of characters in the keyword.

2.5 Automatic Speech recognition

When WUW component detects a predefined keyword, it triggers actual command recording, which is then processed by speech recognition component. In this work, we implemented end-to-end ASR module aiming for offline speech recognition to protect users privacy. In the context of Voice command module, ASR needs to be lightweight and working with limited vocabulary since all command recording and processing should be done on low power embedded device.

The architecture of ASR comprised of three components: Feature extraction part, Acoustic model (AM) responsible for describing distribution over acoustic observations for given the word sequence and the language model (LM) which assigns a probability to every possible word sequence.

Underlying architecture for AM is Bidirectional Recurrent Neural Network with LSTM similar to one employed in [14]. Using bidirectional network allows the output layer to obtain information from past and future states i.e. information about the location of the characters before and after the current utterance can bring improvements in performance but introduces the increase in latency.

This increase in latency is justified considering that role of the ASR in Voice Control module pipeline is speech-to-text transformation where the only accuracy is essential because transformed text command is further process by VCI and user doesn't see actual output form ASR component (there isn't feedback from ASR to the user). Each layer contains 512 LSTM cells (256 neurons per forward and backward sub-layer)

CTC algorithm was used to train sequence to sequence network with two bidirectional LSTM layers. The goal of CTC is to use neural network to transform input sequence into probabilities of characters at each time step, the speaker has spoken 1 of 29 possible characters.

ASR combines sequence score of acoustic model labels over a time frame and sequences score of words from language model and predicts actual speech command as word sequence that maximizes both the language and acoustic model scores. To speed up the training process of ASR in this work we used pretrained *kalidi* [15] LM to estimates the prior probabilities of spoken sounds. AM component is implemented in *Tensorflow* and for

learning parameters we used BPTT with *AdamOptimizer* which is an improved version of traditional gradient descent by using momentum (moving averages of the parameters) to control the learning rate.

2.6 Voice Command Interface

Voice command Interface receives results of the speech-to-text processing as unstructured text data. To create structured input which can be later used by HA to realize user commands, we used the command parser. Command parser accepts the unstructured command text and performs the semantic analysis according to predefined command patterns. Command parser was realized using parse generator ANTLR 4[16] and all possible command patterns are defined in ANTLR grammar file. Before determining command patterns, we need to identify between two types of smart home devices according to their use case: actuator where the user can set state, and the sensor which allows only to get the state. To design commands patterns to control or read the state of an individual device or a group of devices we define three parts: action, device name/device group and location descriptors (optional).

3 Evaluation

Experimental evaluation of the Voice Control module is made by analyzing each component separately. Before the start of the evaluation we asked 20 participants to read a text containing 400 sentences of which 220 were home automation commands. Each command line is preceded by WUW keyword “Elise”. All the recordings are altered by corrupting clean utterances using room simulator, adding varying degrees of noise and reverberation from daily life noisy environmental recording so that overall SNR is between 0db and 30db with average SNR of 12 db. This recorded data set is about 4 hours long and it will be used as our test set.

In this work freely available recordings of transcribed English speech are used as the training set for both WUW and ASR components. Training corpus consists of 1000 hours of speech available as part of *LibriSpeech* dataset [17]. Inputs to RNN acoustic models are 40-dimensional MFCC features and their first and second-order derivates

WUW component is further fine-tuned by adapting the WUW acoustical model to the speakers. Also, we used Speech Commands Dataset [18] containing 65,000 one-second long utterances of 30 short words, by thousands of different people for the ASR model refinement training.

Experimental results for WUW are given regarding false alarm per hour, i.e., false alarm frequency (FAF) which is defined as the incorrect detection of a keyword per hour. False Alarm (FA) rate is calculated by dividing the number of false positives by the total number of examples. False alarm means that the system is activated even though the user did not intend to do so. Reported FAF of the keyword spotter is 0,5. Evaluation results for ASR component are given in terms of word error rate (WER). Reported average WER of *LibriSpeeech* corpus is 21.5% and 32% when the recorded test set is used.

The accuracy of Command Interface depends most on accuracy of speech to text transcription, and it can achieve 100% if the command follows grammar rules.

4 Conclusion

In this work, we presented the architecture of the voice command module that enabled voice control inside the smart home lab. The intent was to build a lightweight module that can run on low power embedded devices like Raspberry Pi. We presented the WUW spotting system that enables user to activate VCM using speech instead of manual activation. ASR was designed to allow offline speech to text processing regarding the matter of privacy. Unstructured text from ASR is parsed by Voice Command Interface and converted to appropriate JSON directive.

Acknowledgment

This work has been fully supported by the Croatian Science Foundation under the project number UIP-2014-09-3875.

References:

[1] De Silva, L.C., Morikawa, C. and Petra, I.M., 2012. State of the art of smart homes. Engineering Applications of Artificial Intelligence, 25(7), pp.1313-1321.

[2] Picone, J., 1996. Fundamentals of speech recognition: A short course. Institute for Signal and Information Processing, Mississippi State University.

[3] Giannakopoulos, T., Tatlas, N.A., Ganchev, T. and Potamitis, I., 2005. A practical, real-time speech-driven home automation front-end.

IEEE Transactions on Consumer Electronics, 51(2), pp.514-523.

[4] McLoughlin, I.V. and Sharifzadeh, H.R., 2007, December. Speech recognition engine adaptions for smart home dialogues. In Information, Communications & Signal Processing, 2007 6th International Conference on (pp. 1-5). IEEE.

[5] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE.

[6] Graves, A., 2012. Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711.

[7] Vinyals, O., Ravuri, S.V. and Povey, D., 2012, March. Revisiting recurrent neural networks for robust ASR. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on (pp. 4085-4088). IEEE.

[8] Li, J., Zhang, H., Cai, X. and Xu, B., 2015. Towards end-to-end speech recognition for chinese mandarin using long short-term memory recurrent neural networks. In Sixteenth annual conference of the international speech communication association.

[9] Schuster, M. and Paliwal, K.K., 1997. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), pp.2673-2681.

[10] Graves, A., 2013. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

[11] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[12] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J., 2006, June. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (pp. 369-376). ACM.

[13] Hwang, K., Lee, M. and Sung, W., 2015. Online keyword spotting with a character-level recurrent neural network. arXiv preprint arXiv:1512.08903.

[14] Graves, A. and Jaitly, N., 2014, January. Towards end-to-end speech recognition with recurrent neural networks. In International Conference on Machine Learning (pp. 1764-1772).

[15] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P. and Silovsky, J., 2011. The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. EPFL-CONF-192584). IEEE Signal Processing Society.

[16] Parr, T., 2013. The definitive ANTLR 4 reference. Pragmatic Bookshelf.

[17] Panayotov, V., Chen, G., Povey, D. and Khudanpur, S., 2015, April. Librispeech: an ASR corpus based on public domain audio books. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 5206-5210). IEEE.

[18] Warden, P., 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv preprint arXiv:1804.03209.