Power Efficient implementation of MVA-SI on Speech Controlled IoT Systems

Conference Paper · May 2016

1 author:

Nisarg Milan Vasavada
Gujarat Technological University
14 PUBLICATIONS **6** CITATIONS

SEE PROFILE

Power Efficient implementation of MVA-SI on Speech Controlled IoT Systems

Nisarg M. Vasavada, Dhwani P. Sametriya, Dipika S. Vasava
Department of VLSI & ESD, GTU PG School, Ahmedabad, India.

***Abstract* — Speech processing is implemented based on application specific accuracy requirements and regardless of the implementation scheme the ultimate goal remains the same, to enhance speech processing engines up to an extent where natural language can be interpreted accurately in case of word or phoneme recognition and context prediction. ASR training and adaption is always application specific and in the considered application the ASR engine is trained for a power efficient IoT application. The nodes communicate via 802.15.4 6LoWPAN which makes wireless sound transmission very lightweight and power efficient. MVA-SI is a customized version of i-vector speaker identification algorithm which identifies the speaker with system wake up call. In this paper, the logic and implementation of MVA-SI are discussed in detail along with elaboration of prototype system architecture.**

Keywords — ASR, MVA-SI, Arduino, 6lowpan, Raspberry Pi

I. INTRODUCTION TO ASR

Automatic Speech Recognition (ASR) is one of the most intuitive methods of interaction with electronic devices which are mostly targeted to be a part of Artificial Intelligence community [1]. Although the output is fundamentally similar to old Speech-to-Text (STT) converters, the recent ASR engines are more context specific, accent neutral and user centric which makes them more efficient and reliable. The basic ASR system consists of a microphone input, a general purpose or dedicated digital signal processor, memory, display device and IO [2]. Many methods, models and algorithms have been developed since then among which many have succeeded and others could only find their space in research documentations and old text books. One of the pioneer algorithms which is still used in most of the speech recognition engines is known as Hidden Markov Model (HMM) [3].

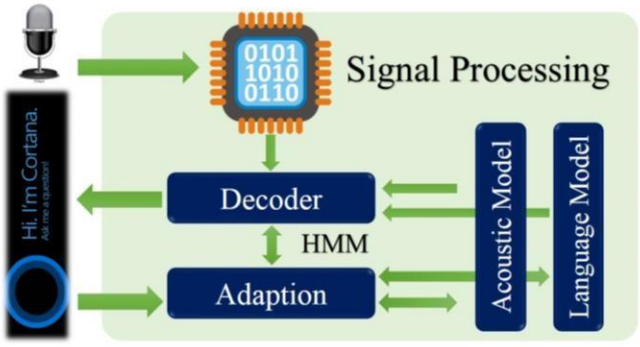


Figure 1 Block diagram of ASR

The project was performed at CDAC ACTS as a part of M.E dissertation at pune.

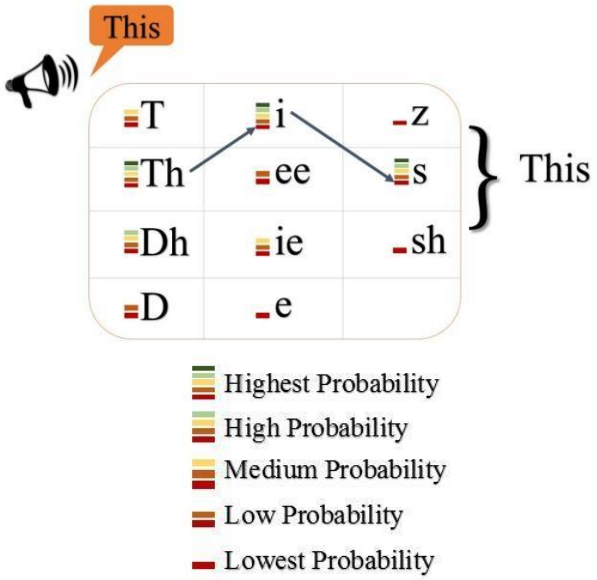


Figure 2: Word Recognition through HMM

The Hidden Markov Model (HMM) is a variant of FSM (Finite State Machine) having discrete hidden states, output state (alphabet), transition probabilities (next alphabet), output (phoneme) probabilities, and initial state probabilities [5]. Instead of having an observable current state it produces outputs with certain probabilities. In the example above, word "This" is broken down in respective phonemes and most probable entries are selected. Other probable utterances with respective priorities are also shown. So, when a user with general accent speaks an utterance starting with "This", mathematically there is also a low probability of him having spoken 'D' instead of 'Th' which depends on frequency created by him but the training of Acoustic model enables ASR to determine the correct phonemes which in turn become key to determine correct word, phrase and eventually the whole utterance. In advanced ASR such as CMU's sphinx, GMM which stands for Gaussian Mixture Model also plays an important role in the STT algorithm by complementing HMM. While HMM finds the probability of each state (in this case, phoneme), GMM finds the probability of weighted sums of various HMM states [10] [12]. This part of ASR is known as Language Model (LM) and depth HMM of GMM depends on the rigor required by the Acoustic Model (AM) and LM. For example, one word utterances may not even need LM while continuous language processors may need more than one layers of LM, for this reason N-gram ASR engines are used since Sphinx2.

II. Low Power IoT

The idea behind the IoT is to make the small and sensor driven embedded devices pervasive and IP (Internet Protocol) enabled, and to integrate them as an essential part of the Internet by making them authentic, area specific and quasi-real- time data suppliers[13]. In an IoT system, data is generated from multiple devices spanning a spectrum of complexities (varying from sensors to servers) which are processed in different ways (varying from small ASICs to sophisticated parallel processors), transmitted to different locations (varying from users to servers), and acted upon by applications (varying from mobile application to enterprise applications). The basic IoT reference model is stacked up of seven levels where each level is defined with its terminology that can be used for standardization to create a globally acceptable frame of reference [13]. As the concept of "Things" has a huge ocean of devices in its own category it is one of the most irrational approaches to treat all of the devices equally. Thus since small sensors and actuators which are embedded devices by nature do not contain scope or requirement of intelligence similar to fully fledged computing systems the TCP/IP layered approach is over-sufficient for such devices and it is efficient to remove wrappers of unwanted layers and implement a scaled down and lightweight protocol [4]. In the figure shown below, 6LoWPAN UDP header is compared to normal 64-bit UDP header. By removing the wrappers of IPv6 addressing the devices are identified by MAC addresses which are unique for every device and is provided by the manufacturer [6]. This allows 127 byte frame to store more payload which in turn gives it flexibility to send more data over the same frame and use less frames for the total amount of data. The 6LoWPAN stands for Low Power IPv6 which means all of the nodes will be a part of IPv6 only and the communication will work hop-hop due to neighborhood discovery [6].

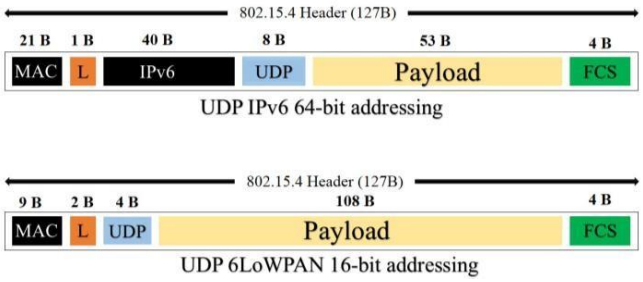


Figure 3: 6LoWPAN Header

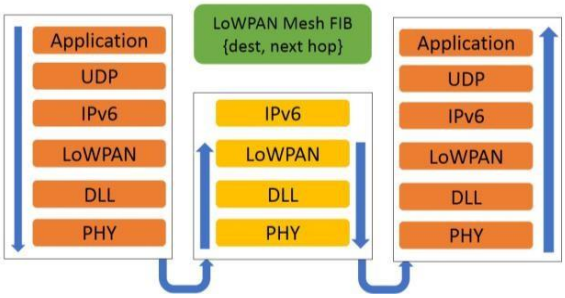
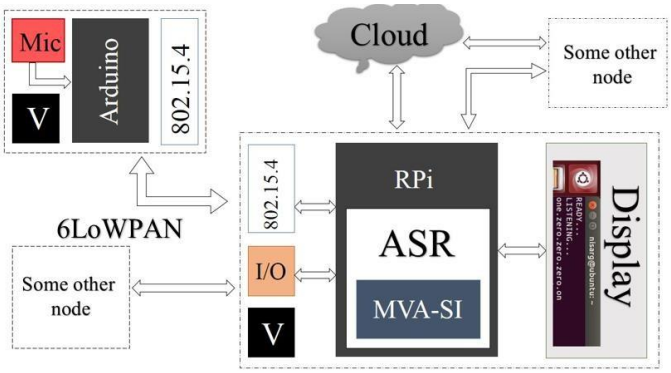
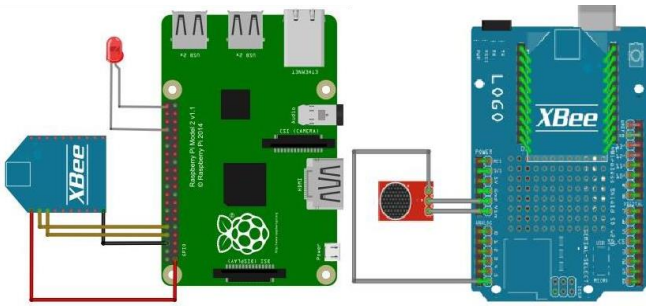


Figure 4: 6LoWPAN Layers

III. Test System Architecture



(a)



(b)

Figure 5: (a) System architecture [11] (b) Gateway node and small node [8] [9]

MVA-SI stands for Modified Vector Algorithm for Speaker Identification which is an efficient solution for optimization of bandwidth, battery life, memory, portability and system cost which is applied on a test system shown in figure (a) [11]. The proposed system is an IoT implementation at constrained embedded level where small nodes connect with gateways using 6LoWPAN. Gateways are provided proper internet over multiple protocols such as IpV6 on network layer and UDP or MQTT (for lightweight UDP) are applied. To communicate with other routers and gateways, TCP or SCTP are applied. The small nodes from 6LoWPAN communicate via hop to hop in personal area so naturally the number if hops are supposed to be limited. Their MAC addresses are stored in the devices only so they do not need to know respective IPv6 IPs. Such a node here is designed using Arduino development board as shown in (c) where mic is connected via GPIO and wireless connectivity is achieved using 802.15.4 xbee radio. On the other hand, Gateway is implemented using Raspberry Pi 2 as shown in (b) with another 802.15.4 xbee radio. It is equipped with CMU Pocketsphinx ASR engine which enables it to recognize the speech utterance provided to Arduino as an input. The generic raspbian Jessie incorporates full featured internet connectivity which can be achieved using Ethernet or Wi-Fi module. The wide range of 1.2km of Xbee radios allow small nodes to remain mobile flexibly whereas larger payload of 6LoWPAN increases data (in Kbps) rate as following.

$$6LoWPAN\ Bitrate = \frac{6LoWPAN\ payload * Xbee\ Bitrate}{Xbee\ Payload}$$

$$6LoWPAN\ Bitrate = \frac{108 * 250}{53} = 519.09$$

IV. MVA – SI

A. *Algorithm and process flow*

In section 1, it was explained how words are recognized using HMM and GMM in ASR. Speaker identification semantically improves the ASR accuracy by reducing WER (Word Error Rate) and creating more context specific relevance. The MVA-SI actually does nothing but analyzing the phones in the speech context more precisely and giving significant eigenvalues for the vectors generated by the input speech. Phones are the starting of consonants and endings of vowels which are all distinct from each other. The algorithm focuses on the phones and the floating point values generated by them and creates an eigenvector. Such eigenvectors are compared to the stored ones and user is recognized. This is all done by one "Hello" wake call which initiates the ASR and also recognizes the user. Since the keyword for this operation is predefined the complexity of the algorithm decreases while the efficiency increases which is the main aim of not just this but ANY system regardless of the domain and purpose. The requirement of such algorithm arises when words are unconventional and mostly not stored in either acoustic or language models. One of such examples is a name. MVA-SI stores the user specific data and provides binary probability to words. This case is different from AM or LM since here either the word is black or white means either the probability is 0 or one. Such a scenario is expressed and resolved mathematically in the expressions below. P1 and P2 are tentative probabilities of utterance whereas 'A' and 'B' are binary probabilities of MVA-SI (It is worth noting that the number of cases may differ for other utterances). P1 and P2 are obtained using trained AM and n-gram LM. Trivially, one will be float and other will be zero so comparison would be easy. Thus Pfinal is derived from MVA-SI.

$$P1 = [P(1)|P(2)] * [P(12)|P(3)] * [P(12 \dots n-1)|P(n)]$$

$$P2 = [P(1)|P(2)] * [P(12)|P(3)] * [P(12 \dots n-1)|P(n)]$$

If P1=P2

$$P1' = P1 * A\ ;\ P2' = P2 * B$$

If P1'>P2'

$$Pfinal = P1'$$

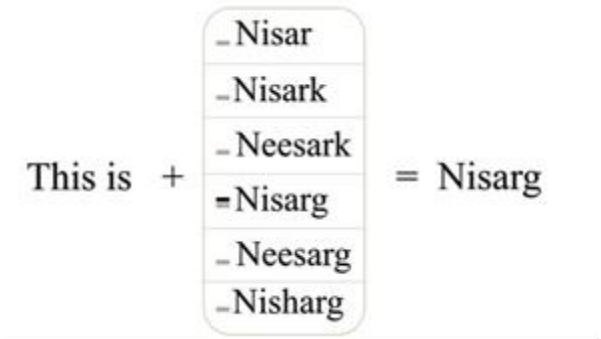


Figure 6: ASR with MVA – SI

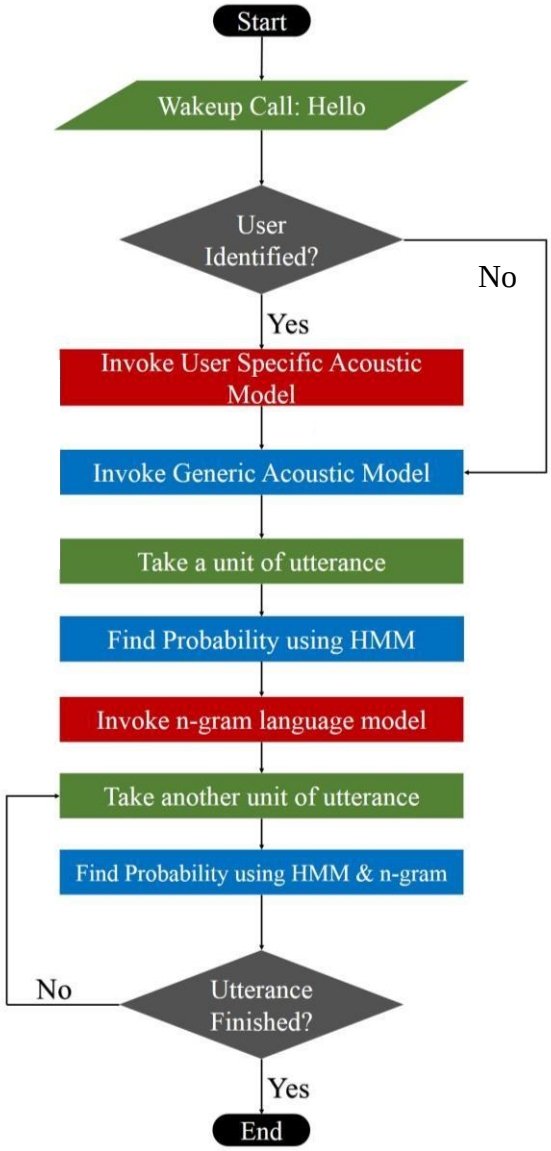


Figure 7: MVA – SI Flowchart

The flowchart elaborates the execution of MVA-SI in above mentioned environment. It is governed by the APIs written in C only. First of all the ASR engine is initiated. The RPi receives sound input from Arduino node which will be seamless as bitrate over 500 is sufficient enough to stream sound without having the buffer partially filled. And since this is the only user enabled key process that will be running at a particular instance the bandwidth will not be shared between the applications. From the "Hello" .wav file, eigenvector is created which, if matches a certain range (provided environmental and natural variance of ±10ve, the pointer of that value will lead to the user member of the user entry union. This is how the user is recognized. We can say that MVA-SI is built on top of i-vector speaker verification algorithm but is a constrained and thus more efficient version of it [14]. Consequently, the AM and LM are invoked and STT

conversion takes place and the process runs in a loop till the utterance is finished. Sphinx STT APIs put EOU character at the end of every array of vectors automatically so it is not user or programmer's concern.

B. Implementation

In figure 8, ASR implementation on Ubuntu Linux is displayed which was performed for testing purpose. It is visible that while trying to perform continuous speech recognition, words that are pronounced without gaps are misinterpreted due to ASR engine being unfamiliar with the accent of the user. On the other hand the same experiment with the same results was performed on RPi2 and afterwards, MVA-SI APIs were added to it. The logic and codes of the APIs are out of the scope of the paper. As it can be seen in figure 9, MVA-SI is running as a separate process and word "hello" was spoken using different accents and different voices which was interpreted accurately. On the other hand 6LoWPAN communication was established between two nodes made of Arduino. To do so, pIPv6 library was imported to Arduino toolkit and serial IPv6 in LoWPAN header APIs were used. The baud rate was kept as default 9600 which is enough to transmit sounds and sampling rate for ASR was kept as 48000. For testing of 6LoWPAN connectivity over Arduino the states of LED were transmitted and it becomes crucial because of the less RAM available in the Arduino which is 2kb only. The difference in results are portrayed in tabular format in conclusion section.

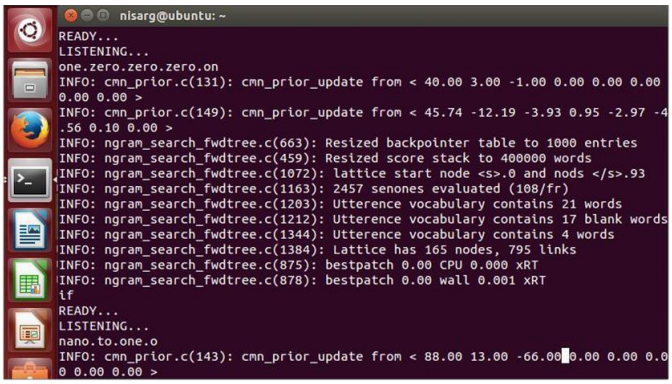


Figure 8: ASR Testing on Ubuntu

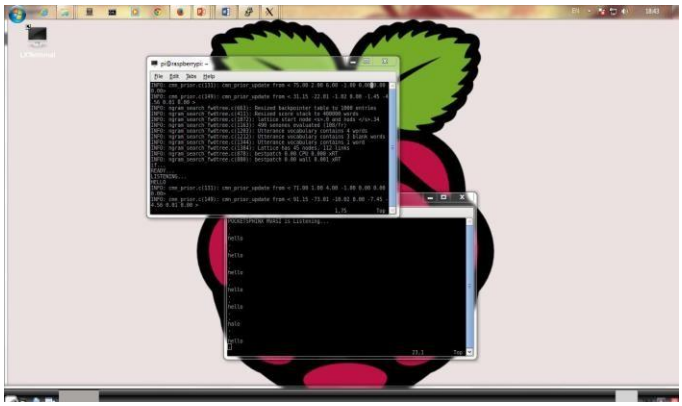


Figure 9: MVA – SI on RPi

Figure 10: 6LoWPAN state transfer

CONCLUSION

Thus by implementing MVA-SI on a power efficient ASR controlled IoT system, it is observed that user specific speech recognition improves with decrement in WER (Word Error Rate). The fact is furnished in the table below.

Table 1: Comparision

Parameter	Existing System	MVA-SI
Power Efficient	No	Yes
WER	23%[7]	8%
Payload on Xbee	53B	108B
Load on Gateway	More	Less
Load on X'mitter	More	Less
802.15.4 Bandwidth	~250kbps[13]u	519Kbbs

REFERENCES

[1] "IoT Reference Model", A white paper by Cisco Inc. Nisarg M. Vasavada, Swapnil Belhe, "A power efficient Scheme for Speech Controlled IoT Applications", IJERT, vol. 5, Issue 1, p.p 446- 449, January 2016. DOI: http://dx.doi.org/10.17577/IJERTV5IS010382

[2] Andrew Kehler et al. "Spoken Language Processing", Prentice Hall New Jersey, ISBN: 978-0131873216.

[3] J.P. Haton, "Speech analysis for automatic speech recognition: A review," Proc. 5-th Conf. on Speech Technology and Human-Computer Dialogue, 2009, vol., no., pp. 1-5, 18-21 June 2009.

[4] Ye-Yi Wang, Dong Yu, Yun-Cheng Ju, and Alex Acero, "An Introduction to Voice Search: A look at the technology, the technological challenges, and the solutions", IEEE Signal Processing Magazine, p.p 29-39, May 2008.

[5] Dhwani P. Sametriya, Nisarg M. Vasavada, "HC-CPSoC: Hybrid Cluster NoC Topology for Cyber-Physical System-on-Chip", 978-1-4673-9338-6/16, IEEE WiSPNET 2016 conference proceedings, p.p 240-243.

[6] M.J.F. Gales, "Acoustic Modelling for Speech Recognition: Hidden Markov Models and Beyond?", IEEE ASRU 2009, p.no 44. Douglas O'Shaughnessy, "Acoustic Analysis for Automatic Speech Recognition", IEEE Proceedings Vol. 101, No. 5, May 2013, pp. 1038-1044

[7] Badamasi Y. A., "The working Principal of an Arduino", 11th International conference on Electronics, Computer and Computing, 2014.

[8] Severence C., "Eben Upton: Raspberry Pi", Computer Conversations by IEEE, p.p 14-16, October 2013.

[9] Willie Walker, Paul Lamere et al., "Sphinx-4: A Flexible Open Source Framework for Speech Recognition" a white paper by Sun Microsystems Inc., 2004.

[10] Guangguang Ma1, Wenli Zhou et al., "A Comparison between HTK and SPHINX on Chinese Mandarin", IEEE Computer Society International joint conference on Artificial Intelligence, p.p 394-397, 2009.

[11] David Huggins-Daines, Mohit Kumar et al., “Pocketsphinx: A free, Real-time continuous Speech recognition system for hand-held devices”, IEEE ICASSP, pp. 185-188, 2006.

[12] Ferdian Thung, Tegawende F. Bissyande et al., "Network Structure of Social Coding in GitHub" IEEE Computer Society 17th European Conference on Software Maintenance and Reengineering, pp. 323-326, 2013.

[13] Wei Li, Tianfan Fu, Jie Zhu, "An improved i-vector extraction algorithm for speaker verification", Springer EURASIP Journal on Audio, Speech, and Music Processing, 2015.

[14] “6LoWPAN: Wireless Embedded Internet”, Z Shelby, C Bormann, Wiley series in communication networking and distributed systems, ISBN: 978-0-470-74799-5.