International Journal of Artificial Intelligence and Neural Networks – IJAINN
Volume 4 : Issue 3 [ISSN 2250 – 3749]
Publication Date : 30 September, 2014

Effect of End Sections in Speech Recognition

Diganta Baishya[1] and Pradip K. Das[2]

***Abstract* - The speech recognition process involves human-machine communication via human voice. This complex process involves understanding and differentiating the basic characteristics of speech. Lot of work has been done in this regard, but we are yet to have a system with hundred percent recognition rate thus limiting uses of such a system only to non-critical applications. The task is to recognize the human voice sent to the machine via a communication media. The paper discusses some works done in the area of speech recognition. All speech recognition algorithms consider specific characteristics of speech signal resulting into better recognition rate in the recent years. However people are still looking into different aspects of speech signal to improve their rates. We focus on an aspect that is not studied much yet. The paper reports experiments conducted to verify whether the position of a word in a sentence influences the recognition rate, and if so, how and what can be done to utilize this effect to improve it. Experiments are conducted placing a word in various positions of a continuously spoken set of words to check the recognition rate. It was observed that the word when placed at the last position of a sequence usually improves the recognition rate. It was interesting to observe that chances of a word being interpreted correctly depend heavily on the phonetic structure towards the end. The observation can be crucial to improvement of the existing algorithms used in speech recognition.**

Keywords - Continuous Speech, HTK, Word Position, Recognition Rate, Viterbi

I. Introduction

Speech recognition process involves human to machine communication via voice signal. In recent years, ability to handle human voice by using technology has undergone a major revolution. Applications and software that are developed to control devices like mobile phones, televisions and computers including laptops to desktops of all sizes are in high demand. Speech recognition systems are now available in most popular languages of the world like English, French, German, Chinese, etc. with accuracy around 90-95% that are capable of recognizing 100-150 words per minute. Even though most of these applications perform well, they are not suitable for real time and critical applications where the margin of error allowed is low. People are still working to improve upon it for more accurate results and to reduce run time cost. This effort has resulted into numerous algorithms.

In his book [1], S. Young states that a speech recognition system assumes the speech signal to be some sequence of messages encoded in the form of some symbols.

[1.] Diganta Baishya
Dept. of CSE, IIT Guwahati, Assam, INDIA

[2.] Pradip K. Das
Dept. of CSE, IIT Guwahati, Assam, INDIA

Hence the speech signal is first converted to equally space discrete parameter vectors, which is assumed to be an exact representation of the speech wave. Any recognizer aims at best possible mapping between speech vectors and symbol sequence. Different symbols may produce similar sounds and they vary a lot speaker-wise which makes the recognition process a difficult task. The other main issue is to recognize the word boundary which is very difficult because of underlying noise. Word boundary is not at all an issue in case of isolated word recognition but in case of continuous word recognition it creates a big challenge.

II. Foundation

Token passing version of Viterbi algorithm [2] views speech recognition as a process of passing tokens around a transition network. In this algorithm, a token is passed from one state to all its neighbouring states in the transition network carrying some probability. The receiving node discards all but the token with the highest probability. The actual path traversed can be retrieved working backwards from the final state node at the end. This algorithm is the basis of the recognition process used in the HTK software.

"CarpeDiem" [3] reduces the transition network for the best path search. To determine the end point of the best path to any layer, we need not inspect all vertices in that layer. The algorithm is based on the concept that "After sorting the vertices in layer *t* according to their vertical weight, the search can be stopped when the difference in vertical weight of the best node so far and the next vertex in the ordering is big enough to counterbalance any advantage that can be possibly derived from exploiting a better transition and/or a better ancestor". This obviously reduces the time taken for search. The running time of an execution of CarpeDiem depends on how the weights of vertical and horizontal features compare: The more discriminative are vertical features with respect to horizontal features, the larger is the edge CarpeDiem has over the Viterbi algorithm.

J Joddel and his colleagues [7] have shown light on reducing the size of the transition network so that the Viterbi algorithm used for recognition needs less amount of time for processing. They also inferred that uncertainty is higher at the beginning of a word than at the end. Hence, using a single fixed beam throughout the word is not a very good idea. It allows one to think of using a word-end beam towards the end of a word for reducing computation. The paper advocates tree structuring the network so that the search time can be improved due to the fact that words sharing the common phonemes at the beginning can share the same model instances, thus reducing the size of the network.

Experiments based on **t**he Viterbi algorithm for ASR with continuous-density HMMs has shown that even if the isolated recognition result is [10] very good, the results for connected digit recognition was poor. However, the important observation they made was that the last digit in the connected sequence was never missed and was perfectly recognized.

The experiments discussed in this paper are based on recognizers developed using the HTK toolkit. A detailed study about the behaviour of the recognizer and the paper published by Rob Oxspring and Mark Greenwood [10] motivated to concentrate on an area which is not explored much.

III. Experiment and Observations

Experiments were conducted to test whether the position of a word can influence its recognition rate. The recognizer was built using HTK and the results found are illustrated below. Crucial observations made thereafter are also listed.

A. *Recordings*

All the recordings are done with a sampling rate of 16000 Hz, 16 bits/sample and with mono channel using the HSLAB utility of HMM Toolkit (HTK). For experiments, minimum ten utterances (a total of 327 utterances) of ten isolated English digits were recorded. The list of digits along with their respective phonetic representation is tabulated below. All recordings are done by a single speaker.

TABLE I. DIGITS WITH THEIR PHONETIC TRANSCRIPTIONS

1.	EIGHT	ey t	7.	FIVE	f ay v
2.	FOUR	f ao	8.	FOUR	f ao r
3.	NINE	n ay n	9.	ONE	w ah n
4.	SEVEN	s eh v n	10.	SIX	s ih k s
5.	THREE	th r iy	11.	TWO	t uw
6.	ZERO	z ia r ow			

.

B. *The Recognizer*

The recognizer to recognize the ten digits was built using the HTK toolkit. The steps followed to design the recognizer are briefly stated in the Appendix.

C. *The Algorithms*

HTK uses a variation of Viterbi Algorithm for recognition. The Viterbi Algorithm is reproduced below [12]:

Initialization (i=0): $v_o(0) = 1$, $v_k(0) = 0$ for $k>0$	$v_o(0) = 1$, $v_k(0) = -\text{lnf}$ for $k>0$
Recursion(i=1…L): $v_l(i) = e_l(x_i) \max_k(v_k(i-1)a_{kl})$ $ptr_i(l) = \text{argmax}_k(v_k(i-1)a_{kl})$	$v_l(i) = e_l(x_i) + \max_k(v_k(i-1)+a_{kl})$ $ptr_i(l) = \text{argmax}_k(v_k(i-1)+a_{kl})$
Termination: $P(x, z^*) = \max_k(v_k(L)a_{ko})$ $z^*_L = \text{argmax}_k(v_k(L)a_{ko})$	$P(x,z^*) = \max_k(v_k(L)+a_{ko})$ $z^*_L = \text{argmax}_k(v_k(L)+a_{ko})$
Trace back (i=L….1): $z_{i-1}^* = ptr_i(z_i^*)$	

The above Algorithm computes the maximum probable state sequence recursively that may throw the observed sequence vectors. But HTK uses a modification of Viterbi Algorithm called Token Passing version of Viterbi Algorithm. It may be stated as follows [13]:

```
Initialization (t=0)
  Initialize each initial state to hold a token with $s = 0$
  All other states are initialized with a token of score $s = -\infty$
Algorithm (t>0):
  Propagate tokens to all possible "next" states
  Prune tokens whose path scores fall below a search beam
Termination (t=T):
  Examine the tokens in all possible final states and find the token with
  the largest Viterbi path score.
  This is the probability of the most likely state alignment
TraceBack:
  At each word boundary, track the recognized word in the list of words
  detected so far.
```

The Token Propagation component uses the following steps:

```
Token Propagation:
for t := 1 to T
    for each state i, do
        Pass token copy in state i to all connecting states j, incrementing s
    end
    for each state i, do
        Find the token in state i with the largest s and discard the rest of
        the tokens in state i. (Viterbi Search)
    end
end
```

D. *Experiment*

Sentences each containing five words were constructed from ten digits so that each digit can be tested at first, mid and last position. The results were observed using "hvite" utility of HTK. Words were spoken very slowly to observe the results using isolated word recognition. Then as we speak the sentences, speed was increased and the recognition rate for continuous recognition was verified.

E. *Results*

1) Isolated Word Recognition

The recognizer was tested by the speaker for all the digits in an isolated manner. All digits were recognized with 100% accuracy.

2) Connected Word Recognition

The recognizer was evaluated with the digits tested at different positions of the digit sequence using the digit strings as given in Table II.

Each Table and Graph below shows the comparative results of each digit being placed in first, mid and end position of sentences respectively.

TABLE II. DIGIT STRINGS USED FOR TESTING

1.	TWO	FOUR	SIX	SEVEN	ZERO
2.	SIX	SEVEN	ZERO	TWO	FOUR
3.	ZERO	TWO	FOUR	SIX	SEVEN
4.	FOUR	SIX	SEVEN	ZERO	TWO
5.	SEVEN	EIGHT	FIVE	THREE	SIX
6.	THREE	ONE	NINE	FIVE	EIGHT
7.	EIGHT	FIVE	ONE	NINE	THREE
8.	FIVE	ONE	EIGHT	THREE	NINE
9.	NINE	EIGHT	THREE	ONE	FIVE
10.	ONE	EIGHT	THREE	SIX	FIVE
11.	FIVE	FOUR	TWO	THREE	ONE

TABLE III. DIGIT RECOGNITION RESULT AT EACH POSITION

Digit	First		Second		Third		Fourth		Fifth	
	Correct	Total	Correct	Total	Correct	Total	Correct	Total	Correct	Total
0	11	24	5	21	11	48	7	20	28	37
1	13	24	24	56	13	26	16	43	25	26
2	21	37	17	24	17	26	20	25	41	41
3	17	29	3	17	30	44	66	79	44	49
4	25	43	30	46	18	24	13	21	25	25
5	18	53	10	26	22	26	13	29	37	44
6	21	25	16	20	27	41	39	48	26	26
7	10	26	8	25	11	37	10	20	18	24
8	14	26	31	70	8	27	6	17	24	29
9	27	41	20	23	20	29	9	26	27	27
Total	**177**	**328**	**164**	**328**	**177**	**328**	**199**	**328**	**295**	**328**

TABLE IV. PERCENT RECOGNITION ACCURACY OF DIGITS AT FIRST, MID AND END POSITIONS

Position / Digits	First	Mid	End
ZERO	45.83	25.84	75.68
ONE	54.17	42.40	96.15
TWO	56.76	72.00	100.00
THREE	58.62	70.71	89.80
FOUR	58.14	67.03	100.00
FIVE	33.96	55.56	84.09
SIX	84.00	75.23	100.00
SEVEN	38.46	35.37	75.00
EIGHT	53.85	39.47	82.76
NINE	65.85	62.82	100.00

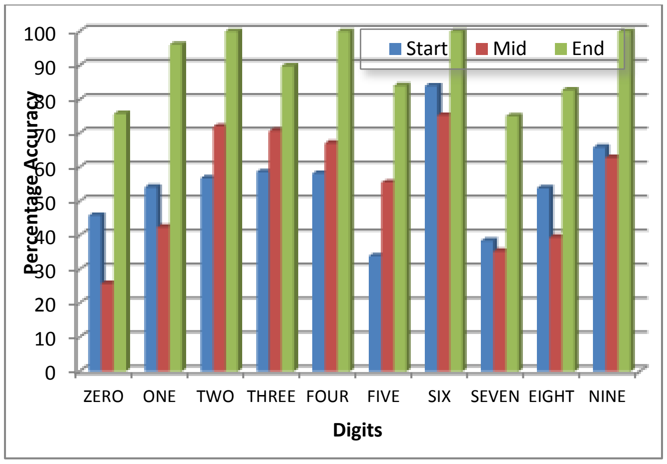

Figure 1. Recognition results for digits at first, mid and end positions.

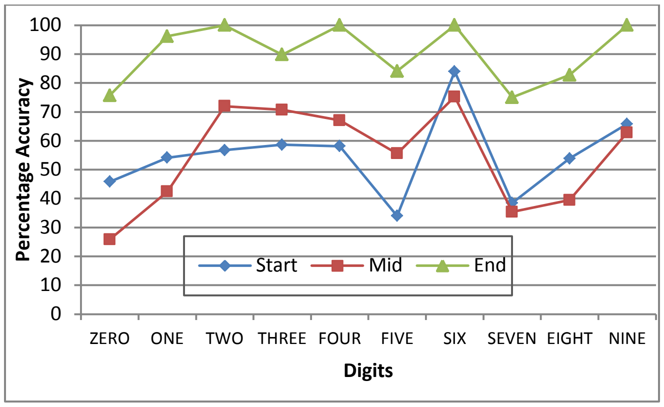

Figure 2. Performance graph of recognition accuracy for digits at start, mid and end positions.

F. *Observations and inferences*

HTK works considerably well for speech recognition. However there is scope of further improvement. In the experiment, the sequences of words were tested by pronouncing very slow initially and then increasing speed. Even though the recognition is 100% for the slow pronunciation, it goes down when the speed is increased. From the above Tables and Figures it is very much evident that the recognition rate is found to be better when the digit is at the last position of a continuously speaking sequence of digits when spoken fast. This leads to some questions:

a. Do we speak the last word of a sentence more carefully?

b. Does the last word spoken is not influenced much by the adjacent word (which is only the preceding word) in comparison to a word in middle of a sentence which is influenced by both its successor and predecessor?

c. Does the Viterbi algorithm perform better in HTK for the last word?

Considering the fact that the last word is pronounced better, this feature may be used to improve the performance of the system and also the last word of a sentence can be used as the key word for some command-based applications. Even one can think of improving the recognition rate for any kind of

International Journal of Artificial Intelligence and Neural Networks – IJAINN
Volume 4 : Issue 3 [ISSN 2250 – 3749]
Publication Date : 30 September, 2014

recognizer. The best will be to reach the end section of a sentence to identify the last word, and then use this information to help searching the best likely word sequence. This method promises to make the searching easy and less time consuming but will require an additional pass to recognize the last digit initially. This may be computed in parallel and the information may be used as a hint in the mid of the original search process for a connected word recognizer.

IV. Conclusions and Future work

In this paper we have presented how the position of a word influences the recognition rate in a continuous sentence and the way we can think of improving the recognition rate using these characteristics. The experiment was based on the significance of the last word. Since the last word is found to be very accurate, we may use it to make the system more accurate with some modifications to the algorithm. Following are the areas that may be focused next:

We may think of designing the commands such that the commands may be implemented as the last word of a sentence.

The recognition algorithm may start with the last word and use this information to reduce the search network which will reduce the size of search network in Viterbi algorithm thus reducing computation time.

Even the recognition process may be initiated in parallel from both directions to reduce the time taken.

V. APPENDIX

Step1: Hparsegram wdnet
Input: The file “gram”. It contains the following text:
$digit = ONE|TWO|THREE|FOUR|FIVE|SIX|SEVEN|EIGHT|NINE|ZERO;
(SENT-START<$digit>SENT-END)
Output: wdnet contains the definition of search network of words to be used for recognition

Step2:HDMan -m -w wlist -n monophones1 -l dlogdict names
Input: wlist, names
Output: monophones1, dlog, dict

Step3: HSgen -l -n 30 wdnetdict>trainprompts
Input: wdnet, dict
Output: trainprompts containing utterances to be trained

Step4: HLEd -l '*' -d dict -i phones0.mlf mkphones0.led train.mlf
Input: dict, mkphones0.led, train.mlf
Output: phones0.mlf

Step5: HCopy -T 1 -C cfg_mfc -S code_mfc.scp
Input: cfg_mfc, code_mfc.scp, Wav files
Output: Corresponding MFC Files
code_mfc.scp contains input directory for wav files and output directory for mfc files

Step6.1: HCompV -C config -f 0.01 -m -S tr_mfc_mono.scp -M hmm0 proto
Input: config, tr_mfc_mono.scp, proto, empty directory “hmm0”
Output: ”macro” and “hmmdefs” in “hmm0”

Step6_2: herest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm0\macros -H hmm0\hmmdefs -M hmm1 monophones0
herest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm1\macros -H hmm1\hmmdefs -M hmm2 monophones0
herest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm2\macros -H hmm2\hmmdefs -M hmm3 monophones0
Input: config, phones0.mlf, tr_mfc_mono.scp,hmm0, empty folders “hmm1”, hmm2” and “hmm3”
Output: “macro” and “hmmdefs” in “hmm1”, hmm2” and “hmm3”

Step6_3: hhed -H hmm4\macros -H hmm4\hmmdefs -M hmm5 sil.hed monophones1
herest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm5\macros -H hmm5\hmmdefs -M hmm6 monophones1
herest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm6\macros -H hmm6\hmmdefs -M hmm7 monophones1
Input: empty directories “hmm5”, “hmm6” & “hmm7” and hmm4 containing files from hmm3 adding parameters for silence model
Output: Output: “macro” and “hmmdefs” in “hmm5”,“hmm6” and “hmm7”

Step6_4: HVite -l * -o swt -b SENT-END -C config -a -H hmm7/macros -H hmm7/hmmdefs -ialigned.mlf -m -t 250.0 -y lab -I train.mlf -S tr_mfc_mono.scpdict, monophones1
Input: hmm7, config, train.mlf, tr_mfc_mono.scp, dict, monophones1
Output: aligned.mlf

Step6_5: herest -C config -I aligned.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm7\macros -H hmm7\hmmdefs -M hmm8 monophones1
herest -C config -I aligned.mlf -t 250.0 150.0 1000.0 -S tr_mfc_mono.scp -H hmm8\macros -H hmm8\hmmdefs -M hmm9 monophones1
Input: config, aligned.mlf, tr_mfc_mono.scp, hmm7, empty directories hmm8 and hmm9
Output: “macro” and “hmmdefs” in “hmm8”,“hmm9”

Step7_1: HLED -n op\triphones1 -l '*' -i op\wintri.mlfmktri.ledaligned.mlf
Input: mktri.led and aligned.mlf
Output: triphones1 and wintri.mlf

Step7_2: HHEd -B -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.hed monophones1
Input: hmm9, Empty directory “hmm10”, mktri.hed and monophones1
Output: “macro” and “hmmdefs” in “hmm10”

Step7_3: herest -B -C config -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S tr_mfc_mono.scp -H hmm10\macros -H hmm10\hmmdefs -M hmm11 triphones1
herest -B -C config -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S tr_mfc_mono.scp -H hmm11\macros -H hmm11\hmmdefs -M hmm12 triphones1
herest -B -C config -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S tr_mfc_mono.scp -H hmm12\macros -H hmm12\hmmdefs -M hmm13 triphones1
Input: config, wintri.mlf, tr_mfc_mono.scp, ”hmm10”,empty directories ”hmm11”, ”hmm12” and ”hmm13”
Output: “macro” and “hmmdefs” in “hmm13”

Step7_4: HVite -H hmm13/macros -H hmm13/hmmdefs -C config2 -w wdnet -p 0.0 -s 5.0 dict triphones1
Input: hmm13, config2, wdnet, dict and triphones1
Output: Recognition Result

VI. REFERENCES

[1] Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason,Valtcho Valtchev and Phil Woodland :The “HTK Book” (for HTK Version 3.1), Cambridge University Engineering Department, December 2001.
[2] S J Young, N H Russel and J H S Thornton, “Token Passing-a simple Conceptual Model for Connected Speech Recognition Systems”, Cambridge University Engineering Department Technical Report, Tech. Report No 38, July 31, 1989.
[3] Roberto Esposito and Daniele P. Radicioni, “CarpeDiem: Optimizing the Viterbi Algorithm and Applications to Supervised Sequential

Learning" published in Journal of Machine Learning Research in August 2009, pp. 1851-1880.

[4] Mohit Dua, R.K.Aggarwal, Virender Kadyan and Shelza Dua,"Punjabi Automatic Speech Recognition Using HTK", published in International Journal of Computer Science Issues, Vol. 9, Issue 4, No 1, July 2012, pp. 359-364.

[5] Preeti Saini, Parneet Kaur and Mohit Dua, "Hindi Automatic Speech Recognition using HTK", Published in International Journal of Engineering Trends and Technology (IJETT) –Volume 4 Issue 6, June 2013, pp 2223-2229.

[6] P. Vijai Bhaskar, Prof. Dr. S. Rama Mohan Rao and A.Gopi, "HTK based Telugu Speech Recognition", published in International Journal of Advanced Research in Computer Science and Software Engineering, December 2012, pp. 307-314.

[7] J Joddel, V Valtchev and S. J Young, "A One Pass Decoder for large Vocabulary Recognition", Proceedings of HLT-94 (Workshop on Human Language Technology), pages 405-410.

[8] K M Knill and S J Young, "Fast Implementation Methods for Viterbi based Word Spotting", Published in Proceedings of the Acoustics, Speech and Signal Processing, 1996, IEEE International Conference - Volume 0, Pages 522-525.

[9] Shay Mozes, Oren Weimann and Michal Ziv-Ukelson, "Speeding Up HMM Decoding and Training by Exploiting Sequence Repetitions", Published in Proceedings of CPM'07 (Proceedings of the 18th Annual Conference on Combinatorial Pattern Matching, pages 4-15.

[10] Rob Oxspring, Mark Greenwood**, "**The Viterbi Algorithm for ASR with Continuous-Density HMMs", University of Sheffield Technical Report, available in:
http://www.dcs.shef.ac.uk/~mark/uni/speech2.pdf

[11] Lilie Mundalifah Roosman, "Phonetic Experiments on the Word and Sentence Prosody of Betawi Malay and Toba Batak", Published by LOT, Netherland, ISBN-10: 90-76864-98-5, ISBN-13: 978-90-76864-98-3.

[12] J Joddel, V Valtchev, Dave Ollason, Phil Woodland, HTK Book (for Version 2.1), March 1997, available in:
http://www.ee.columbia.edu/ln/LabROSA/doc/

[13] Prof. Bryan Pellom, "Automatic Speech Recognition: From Theory to Practice", Class notes of Department of Computer Science, Centre for Spoken Language Research, University of Colorado, available in: http://www.cs.tut.fi/~puhtunn/

About the Author (s):

Diganta Baishya is a final year Graduate student from the Department of Computer Science & Engineering at IIT Guwahati. He is working on speech recognition targetted towards real-time implementation on resource contrained devices.

Pradip K. Das is a faculty member in the Department of Computer Science & Engineering at IIT Guwahati. His research interests include, Speech Processing, Man-Machine interfaces, Mobile Robotics, algorithms, etc.