Evaluating the Performance of a Speech Recognition based System

Vinod Kumar Pandey, Sunil Kumar Kopparapu

TCS Innovation Labs - Mumbai,
Tata Consultancy Services, Pokharan Road 2
Thane 400 601, Maharastra, India.
{vinod.pande, sunilkumar.kopparapu}@tcs.com

Abstract. Speech based solutions have taken center stage with growth in the services industry where there is a need to cater to a very large number of people from all strata of the society. While natural language speech interfaces are the talk in the research community, yet in practice, menu based speech solutions thrive. Typically in a menu based speech solution the user is required to respond by speaking from a closed set of words when prompted by the system. A sequence of human speech response to the IVR prompts results in the completion of a transaction. A transaction is deemed successful if the speech solution can correctly recognize all the spoken utterances of the user whenever prompted by the system. The usual mechanism to evaluate the performance of a speech solution is to do an extensive test of the system by putting it to actual *people use* and then evaluating the performance by analyzing the logs for successful transactions. This kind of evaluation could lead to dissatisfied test users especially if the performance of the system were to result in a poor transaction completion rate. To negate this the Wizard of Oz approach is adopted during evaluation of a speech system. Overall this kind of evaluations is an expensive proposition both in terms of time and cost. In this paper, we propose a method to evaluate the performance of a speech solution without actually putting it to *people use*. We first describe the methodology and then show experimentally that this can be used to identify the performance bottlenecks of the speech solution even before the system is actually used thus saving evaluation time and expenses.

Keywords: Speech solution evaluation, Speech recognition, Pre-launch recognition performance measure.

1 Introduction

There are several commercial menu based ASR systems available around the world for a significant number of languages and interestingly speech solution based on these ASR are being used with good success in the Western part of the globe [5], [3], [6], [2]. Typically, a menu based ASR system restricts user to

speak from a pre-defined closed set of words for enabling a transaction. Before commercial deployment of a speech solution it is imperative to have a quantitative measure of the performance of the speech solution which is primarily based on the speech recognition accuracy of the speech engine used. Generally, the recognition performance of any speech recognition based solution is quantitatively evaluated by putting it to actual use by the people who are the intended users and then analyzing the logs to identify successful and unsuccessful transactions. This evaluation is then used to identifying any further improvement in the speech recognition based solution to better the overall transaction completion rates. This process of evaluation is both time consuming and expensive. For evaluation one needs to identify a set of users and also identify the set of actual usage situations and perform the test. It is also important that the set of users are able to use the system with ease meaning that even in the test conditions the performance of the system, should be good, while this can not usually be guaranteed this aspect of keeping the user experience good makes it necessary to employ a wizard of Oz (WoZ) approach. Typically this requires a human agent in the loop during actual speech transaction where the human agent corrects any mis-recognition by actually listening to the conversation between the human user and the machine without the user knowing that there is a human agent in the loop. The use of WoZ is another expense in the testing a speech solution. All this makes testing a speech solution an expensive and time consuming procedure.

In this paper, we describe a method to evaluate the performance of a speech solution without actual people using the system as is usually done. We then show how this method was adopted to evaluate a speech recognition based solution as a case study. This is the main contribution of the paper. The rest of the paper is organized as follows. The method for evaluation without testing is described in Section 2. In Section 3 we present a case study and conclude in Section 4.

2 Evaluation without Testing

Fig. 1 shows the schematic of a typical menu based speech solution having 3 nodes. At each node there are a set of words that the user is expected to speak and the system is supposed to recognize. In this particular schematic, at the entry node the user can speak any of the n words, namely W_1 or W_2 or $\cdots$ or W_n; n is usually called the perplexity of the node in the speech literature. The larger the n the more the perplexity and higher the confusion and hence lower the recognition accuracies. In most commercial speech solutions the perplexity is kept very low, typically a couple of words. Once the word at the entry node has been recognized (say word W_k has been recognized), the system moves on to the second node where the active list of words to be recognized could be one of W_{k1}, W_{k2}, W_{k3}, ... W_{kp} if the perplexity at the k^{th} node is p. This is carried on to the third node. A transaction is termed successful *if and only if* the recognition at each of the three nodes is correct. For example, typically in a banking speech solution the entry node could expect someone to speak among

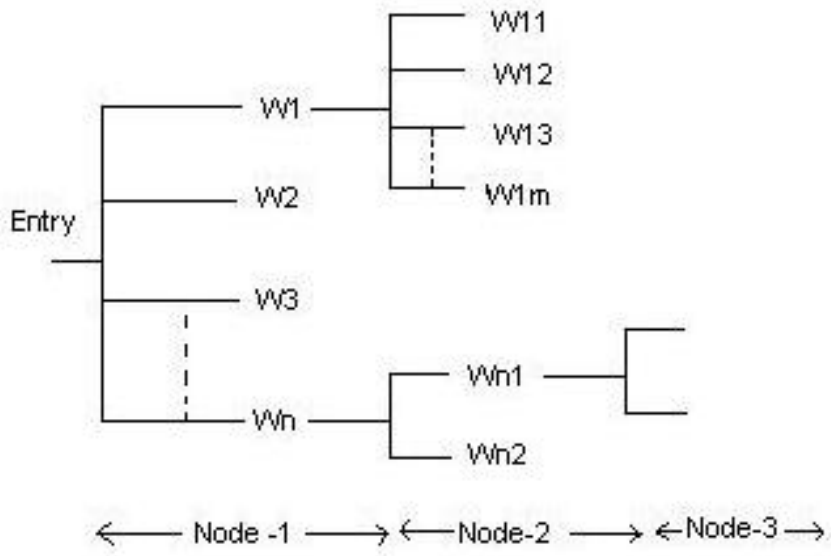

Fig. 1. Schematic of a typical menu based ASR system (W_n is spoken word).

/credit card/[1], */savings account/*, */current account/*, */loan product/*, */demat/*, and */mutual fund transfer/* which has a perplexity of 6. Once a person speaks, say, */savings account/* and is recognized correctly by the system, at the second node it could be */account balance/* or */cheque/* or */last* 5 *transactions/* (perplexity 3) and at the third node (say, on recognition of */cheque/*) it could be */new cheque book request/*, */cheque status/*, and */stop cheque request/* (perplexity 3).

Note 1. Though we will not dwell on this, it is important to note that an error in recognition at the entry node is more expensive than a recognition error at a lower node.

Based on the call flow, and the domain the system can have several nodes for completion of a transaction. Typical menu based speech solutions strive for a 3 - 5 level nodes to make it usable. In any speech based solution (see Fig. 2) first the spoken utterance is hypothesized into a sequence of phonemes using the acoustic models. Since the phoneme recognition accuracy is low, instead of choosing one phoneme it identifies l-best (typically $l = 3$) matching phonemes. This phone lattice is then matched with all the expected words (language model) at that node to find the best match. For a node with perplexity n the constructed phoneme lattice of the spoken utterance is compared with the phoneme sequence representation of all the n words (through the lexicon which is one of he key

[1] We will use / / to indicate the spoken word. For example $/W_1/$ represents the spoken equivalent of the written word W_1.

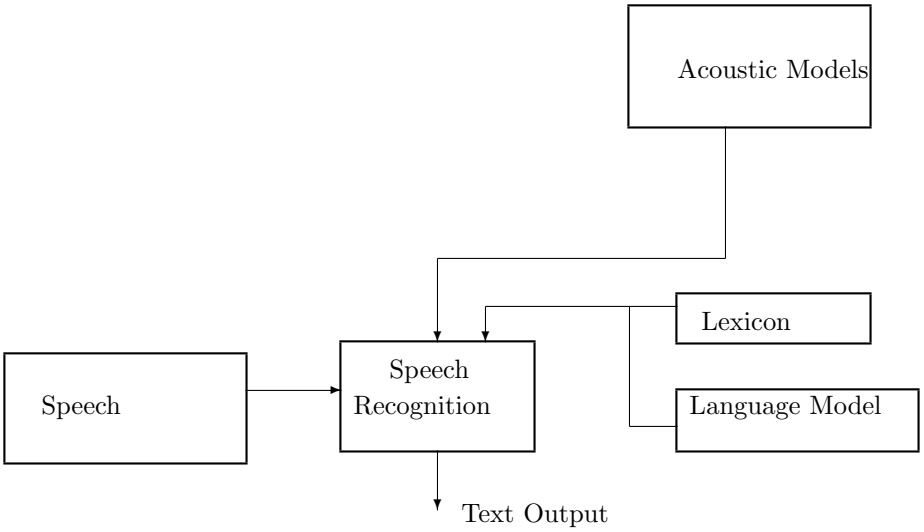

Fig. 2. A typical speech recognition system. In a menu based system the language model is typically the set of words that need to be recognized at a given node.

components of a speech recognition system). The hypothesized phone lattice is declared one of the n words depending on the closeness of the phoneme lattice to the phoneme representation of the n words.

We hypothesize that we can identify the performance of a menu based speech system by identifying the possible confusion among all the words that are active at a given node.

Note 2. If active words at a given node are phonetically similar it becomes difficult for the speech recognition system to distinguish them which in turn leads to recognition errors.

We used Levenshtein distance [1], [4] a well known measure to analyze and identify the confusion among the active words at a given node. This analysis gives a list of all set of words that have a high degree of confusability among them; this understanding can be then used to (a) restructure the set of active words at that node and/or (b) train the words that can be confused by using a larger corpus of speech data. This allows the speech recognition engine to be equipped to be able to distinguish the confusing words better. Actual use of this analysis was carried out for a speech solution developed for Indian Railway Inquiry System to identify bottlenecks in the system before its actual launch.

3 Case Study

A schematic of a speech based Railway Information system, developed for Hindi language is shown in Fig. 3. The system enables user to get information on five

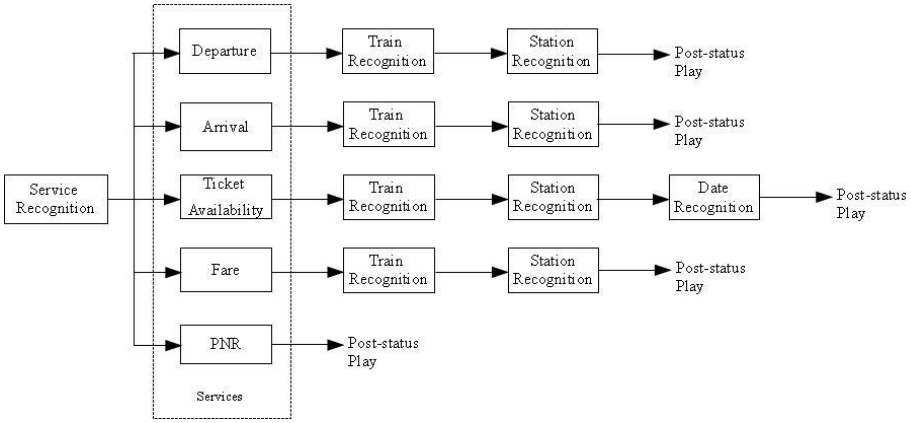

Fig. 3. Call flow of Indian Railway Inquiry System (W_n is spoken word)

different services, namely, (a) ARRIVAL of a given train at a given station, (b) DEPARTURE of a given train at a given station, (c) TICKET AVAILABILITY on a given date in a given train between two stations, and class, (d) FARE in a given class in a given train between two stations, and (e) PNR STATUS. At the first recognition node (node-1), there are one or more active words corresponding to each of these services. For example, for selecting the service FARE, the user can speak among */kiraya jankari/*, */kiraya/*, */fare/*. Similarly, for selecting service TICKET AVAILABILITY, user can speak */upalabdhata jankari/* or */ticket availability/* or */upalabdhata/*.

Note 3. Generally the perplexity at a node is greater than on equal to the number of words that need to be recognized at that node.

In this manner each of the services could have multiple words or phrases that can mean the same thing and the speaker could utter any of these words to refer to that service. The sum of all the possible different ways in which a service can be called (d_i) summed over all the 5 services gives the perplexity ($\mathcal{N}$) at that node, namely,

$$\mathcal{N} = \sum_{i=0}^{5} d_i \tag{1}$$

The speech recognition engine matches the phoneme lattice of the spoken utterance with all the $\mathcal{N}$ words which are active. The active word (one among the $\mathcal{N}$ words) with highest likelihood score is the recognized word. In order to avoid low likelihood recognitions a threshold is set so that even the best likelihood wordis returned only if the likelihood score is greater than the predefined threshold. Completion of a service requires recognitions at several nodes with different perplexity at each node. Clearly depending on the type of service that the user is wanting to use; the user has to go through different number of recognition nodes. For example, to complete the ARRIVAL service it is required to

pass through 3 recognition nodes namely (a) selection of a service, (b) selection of a train name and (c) selection of the railway station. While the perplexity (the words that are active) at the service selection node is fixed the perplexity at the station selection node could depend on the selection of the train name at an earlier node. For example, if the selected train stops at 23 stations, then the perplexity at the station selection node will be ≥ 23.

For confusability analysis at each of the node, we have used the Levenshtein distance [4] or the edit distance as is well known in computer science literature. We found that the utterances /*Sahi*/ and /*Galat*/ have 100% recognition. These words Sahi is represented by the string of phonemes in the lexicon as S AA HH I and the word Galat is represented as the phoneme sequence G L AX TT in the lexicon. We identified the edit distance between these two words Sahi and Galat and used that distance measure as the threshold that is able to differentiate any two words (say T). So if the distance between any two active words at a given recognition node is lower than the threshold T, then there is a greater chance that those two active words could get confused (one word could be recognized as the other which is within a distance of T). There are ways in which this possible misrecognition words could be avoided. The easiest way is to make sure that these two words together are not active at a given recognition node.

Table 1. List of Active Words at node 1

W. No.	Active Word	Phonetic
W_1	kiraya_jankari	K I R AA Y AA J AA tN K AA R I
W_2	kiraya	K I R AA Y AA
W_3	fare	F AY R
W_4	aagaman_jankari	AA G AX M AX tN J AA tN K AA R I
W_5	aagaman	AA G AX M AX tN
W_6	arrival_departure	AX R AA I V AX L dD I P AA R CH AX R
W_7	upalabdhata_jankari	U P AX L AX B tDH AX tT AA J AA tN K AA R I
W_8	ticket_availability	tT I K EY tT AX V AY L AX B I L I tT Y
W_9	upalabdhata	U P AX L AX B tD AX tTH AA
W_{10}	arrival	AX R AA I V AX L
W_{11}	prasthan	P R AX S tTH AA tN
W_{12}	departure	dD I P AA R CH AX R
W_{13}	p_n_r_jankari	P I EY tN AA R J AA tN K AA R I
W_{14}	p_n_r	P I AX tN AA R

Table 1 shows the list of active word at the node 1 when the speech application was initially designed and Table 2 shows the edit distance between all the active words at the node service given in Fig. 3. The distance between words *Sahi* and *Galat* was found to be 5.7 which was set at the threshold, namely $T = 5.7$. This threshold value was used to identify confusing active words. Clearly, as seen

in the Table the distance between word pairs *fare*, *pnr* and *pnr*, *prasthan* is 5.2 and 5.8 respectively, which is very close to the threshold value of 5.7. This can cause a high possibility that */fare/* may get recognized as *pnr* and vice-versa. One can derive from the analysis of the active words that *fare* and *pnr* can

Table 2. Distance Measurement for Active Words at Node 1 of the Railway Inquiry System

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}
W_1	0	8.4	14.2	8.5	14.1	15.3	11	20	17.1	13	13	13.2	6.2	11.2
W_2	8.4	0	7.2	13.8	8.5	14.7	17.8	15.7	11	7.2	7.8	7.7	11.2	6.2
W_3	14.2	7.2	0	14.2	7.2	14.8	18.2	15.8	11.2	8.2	8.2	7.8	14.2	**5.8**
W_4	8.5	13.8	14.2	0	8.4	15.9	9.6	19.7	14.3	13	13	14.7	8.2	11.2
W_5	14.1	8.5	7.2	8.4	0	13.2	16.7	15.7	9.7	7.2	6.7	8.2	14.1	7.1
W_6	15.3	14.7	14.8	15.9	13.2	0	18.7	18.9	15.5	9.4	13.7	7	17.3	11.8
W_7	11	17.8	18.2	9.7	16.7	18.7	0	20	11.2	17.1	15.6	17.8	10.2	13.8
W_8	20	15.7	15.8	19.7	15.7	18.9	20	0	14.5	15.8	17.5	16.5	18.6	15.7
W_9	17.1	11	11.2	14.3	9.7	15.5	11.2	14.5	0	10	9.2	11.1	16.3	9.7
W_{10}	13	7.2	8.2	13.1	7.2	9.4	17.1	15.8	10	0	8.5	8	13	7.8
W_{11}	13.1	7.8	8.2	13.1	6.7	13.7	15.7	17.5	9.2	8.5	0	8.4	11.7	**5.2**
W_{12}	13.2	7.7	7.8	14.7	8.2	7	17.8	16.5	11.1	8.1	8.4	0	12.1	6.8
W_{13}	6.2	11.2	14.2	8.2	14.1	17.3	10.2	18.6	16.3	13.1	11.7	12.1	0	9.8
W_{14}	11.2	6.2	**5.8**	11.2	7.1	11.8	13.8	15.7	9.6	7.8	**5.2**	6.8	9.8	0

not coexist as active words at the same node. The result of the analysis was to remove the active words *fare* and *pnr* at that node.

When the speech system was actually tested by giving speech samples, 17 out of 20 instances of */pnr/* was was recognized as *fare* and vice-versa. Similarly 19 out of 20 instances */pnr/* was misrecognized as *prasthan* and vice versa This confusion is expected as can be seen from the edit distance analysis of the active words in the Table 2. This modified active word list (removal of *fare* and *pnr*) increased the recognition accuracy at the service node (Fig. 3) by as much as 90%.

A similar analysis was carried out at other recognition nodes and the active word list was suitably modified to avoid possible confusion between active word pair. This analysis and modification of the list of active words at a node resulted in a significant improvement in the transaction completion rate. We will present more experimental results in the final paper.

4 Conclusion

In this paper we proposed a methodology to identify words that could lead to confusion at any given node of a speech recognition based system. We used

edit distance as the metric to identifying the possible confusion between the active words. We showed that this metric can be used effectively to enhance the performance of a speech solution without actually putting it to people test. There is a significant saving in terms of being able to identify recognition bottlenecks in a menu based speech solution through this analysis because it does not require actual people testing the system. This methodology was adopted to restructuring the set of active words at each node for better speech recognition in an actual menu based speech recognition system that caters to masses.

References

1. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press (2001)
2. Kim, C., Stern, R.: Feature extraction for robust speech recognition based on maximizing the sharpness of the power distribution and on power flooring. IEEE International Conference on Acoustics Speech and Signal Processing pp. 4574–4577 (2010)
3. Lua, X., Matsudaa, S., Unokib, M., Nakamuraa, S.: Temporal contrast normalization and edge-preserved smoothing of temporal modulation structures of speech for robust speech recognition. Speech Communication 52, 1–11 (2010)
4. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys 33, 31–88 (2001)
5. Sun, Y., Gemmeke, J., Cranen, B., Bosch, L., Boves, L.: Using a dbn to integrate sparse classification and gmm-based asr. Proceedings of Interspeech 2010 (2010)
6. Zhao, Y., Juang, B.: A comparative study of noise estimation algorithms for vts-based robust speech recognition. Proceedings of Interspeech 2010 (2010)