DIRECTORY RETRIEVAL USING VOICE FORM-FILLING

S. Parthasarathy

AT&T Labs Research, Florham Park, NJ, USA

A. Moreno-Daniel

Center for Signal and Image Processing
Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT

Accurate retrieval of entries from large directories is a difficult task. Practical systems attempt to achieve acceptable performance using dialog to restrict the size of the directory. For instance, knowledge of city and state can be used to restrict the entries in a telephone number retrieval application. It is shown that it is advantageous to use a voice form-filling paradigm in which the user speaks all the field entries, first name, last name, city, and state, in a single utterance. A two-pass method for *form-filling* presented recently [1] is evaluated on the directory retrieval task. A delayed network expansion and pruning method is proposed to improve the efficiency of short-list generation in the form-filling algorithm. Experimental results demonstrate that sentence accuracies greater than 85% can be achieved on directory sizes of upto 8 million entries, with modest computing requirements.

Index Terms— Speech recognition, information retrieval, voice form filling

1. INTRODUCTION

Accurate recognition of fields such as people's names or addresses is difficult because the vocabularies can be very large. In addition, the recognition accuracy for each field has to be very high for the overall *form-accuracy* to be acceptable. Practical systems attempt to achieve acceptable performance using dialog to restrict the size of the directory [2, 3]. In a typical telephone directory application, location information such as the city and state are used to restrict the entries to be recognized. In such systems, since the cost of mis-recognizing the location is very high, confirmation of the user's response at each stage of the dialog is important. Further, if the user does not know the response to a particular field such as the city, it takes one or more dialog turns for the system to determine that information. One way of avoiding such cumbersome dialog is to allow the user to speak all the relevant fields necessary to obtain a specific entry in a single utterance.

An effective and scalable method for *form-filling* by voice (vForms) was presented recently to address this problem [1]. In this method, *inter-field* constraints are exploited to improve ASR accuracy. Simple methods for incorporating these constraints include the construction of a grammar for the complete form, or dynamically constructing grammars for each field constrained by input already provided by the user. These simple methods can get impractical for forms with many fields and large vocabularies [2]. The novelty of vForms is the use of a two-pass scheme to achieve (a) high form-accuracy, (b) scalability to large vocabularies, and (c) a flexible user interface that allows the user to speak all the fields of interest in a single utterance.

The goal of this work is to evaluate the performance of vForms on a directory retrieval task and propose ideas for improving the performance of the index access scheme which is at the core of vForms. The basic algorithm is summarized in Sec. 2. A description of the task and implementation details are presented in Sec. 3. Experimental results and analysis are covered in Sec. 4.

2. VFORMS: VOICE FORM-FILLING

An *entry* in a *form* is a collection of a number of *fields*. Constraints that apply across these fields, implicitly specified by the database of form-entries, can be exploited to improve ASR accuracy. Incorporating these constraints in a conventional one-pass speech recognizer becomes impractical for large databases such as a national listing of first name, last name, city, and state [2]. A two-pass form-filling algorithm first introduced in [1] is described below.

1. **Index generation:** Create an index of valid form-entries using *index-terms* such as phone n-grams. The first step consists of converting each entry e_i into a *verbal* form. The verbal form incorporates transformations that help maximize coverage of user utterances. The verbal form of entry i is represented by a phone lattice L_i. Then, a transducer T_i is constructed that associates the entry index i with each term t (diphone, triphone, etc.) that appears in L_i. The final index

$$\mathcal{I} = \mathsf{Det}_{\mathsf{log}} \left(T_1 \cup T_2 \cup \ldots \cup T_N \right) \qquad (1)$$

where $\cup$ represents the union of transducers, and $\mathsf{Det}_{\mathsf{log}}$ refers to determinization in the log semiring [4]. This step is performed off-line and the index needs to be updated only when new entries are added.

2. **First-pass recognition:** Obtain a phone-lattice R using an n-gram phonotactic grammar. The database used to train the task-specific phonotactic model is often the same collection of phone lattices, $\{L_1, L_2, \ldots, L_N\}$, used to train the index. The n-gram model is trained using conventional language modeling tools [5].

3. **Short-list generation:** Query the database using the phone lattice result generated in the first-pass and generate a shortlist of possible database entries. The various steps are listed below. For more details see [1].
   • The costs in R are normalized such that the best path cost is 0.
   • For efficiency, only the terms that are used in the index are retained to produce a cost-normalized, query

$$Q = \mathsf{Det}_{\log}[\Pi_o(R \circ T_t)] \tag{2}$$

   where $\circ$ represents composition of transducers, Π_o represents projection to output symbols and T_t is a transducer that retains only the terms of interest.
   • The list of entry-indices that contain the terms in Q and the associated cost is given by

$$I^1 = \Pi_o(Q \circ \mathcal{I}) \tag{3}$$

   The n-best lowest cost indices, computed by $\oplus_{\log}$ over all terms, are given by

$$I^s = \mathcal{B}_{\text{trop}}[\mathsf{Det}_{\log}(I^1)] \tag{4}$$

   where $\mathcal{B}_{\text{trop}}$ represents the bestpath operation [4].

4. Rescore the shortlist using all available lexical and inter-field constraints to get the final result.

3. IMPLEMENTATION ISSUES

Each step of the algorithm described in Sec. 2 can be optimized in a number of ways. Some of these optimization ideas are explored in this section.

3.1. Task description

Each entry in the database consists of four fields: first name, last name, city, and state. The query consists of utterances in which the user spoke all four fields. In order to demonstrate the scalability of vForms, experiments were performed using database sizes of 250 thousand, 4 million, and 8 million entries. The baseline system is a conventional one-pass fully-constrained (FC) recognizer in which the grammar accepts only valid entries. The baseline results were obtained using optimized networks [6] and represent the best results achieved on this database.

Some statistics of the database are shown in Table 1. The distribution of index terms across form entries is very skewed.

# entries	vocab	n-gram	# terms
250 K	85 K	3	29 K
250 K	85 K	4	228 K
4 M	400 K	3	40 K
4 M	400 K	4	511 K

Table 1. Statistics of terms and entries for a directory retrieval task.

Terms that straddle a field boundary appear only in a few entries and are very informative. Others appear in many form entries. A cumulative distribution of the number of entries in which a particular term appears is shown in Fig. 1 for an index with $4M$ entries and about $40K$ terms. It is clear that most terms appear in only a few entries, but there is at least one term that appears in about $600K$ entries. If such a term is triggered by the query, a large number of paths will be expanded in Eq. 3. This limits scalability of the algorithm to very large directories. Pruning techniques for minimizing the number of entries that need to be processed are described in Sec. 3.2.

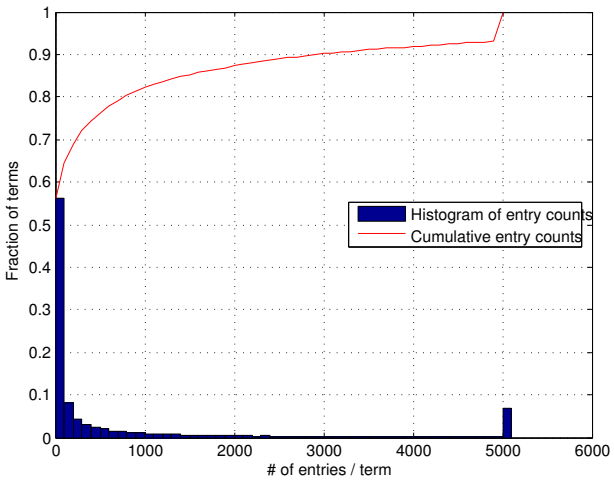

Fig. 1. Histogram of terms for the $4M$ entry index. The value of the last bin is the count of all terms that appear in > 5000 entries.

3.2. Short-list generation

Efficiency of Step 3 of vForms can be improved using a 2-stage expansion strategy which involves delayed expansion of entries as described below. One way of measuring the effectiveness of pruning is to count the number of different entries that are expanded during short-list generation in Eq. 3.

The i-th arc in query Q (Eq. 3) contains the tuple (t_i, e_i, c_i), where t_i is a term that appears in e_i with an expected cost c_i.

Costs are defined as negative of log-likelihood, and the expected cost c_i for a given term t_i is the sum (in the log semiring) of the cost of all paths that contain the term t_i. The least cost (highest likelihood) terms are expanded first. At any stage of term expansion, an active set of entries $S_e = \{e_1, e_2, \ldots, e_N\}$ are maintained ordered by the cumulative cost

$$C_{e_i} = -\log\left[\sum_{t_k \in S(e_i)} \exp^{-c_i}\right] \quad (5)$$

where the sum is over the subset $S(e_i)$ of terms expanded thus far which appear in the entry e_i. Entries with $C_{e_i} > \tau$ are pruned. The value of τ is empirically set to be a certain fixed value above the lowest cumulative cost. Entry pruning is likely to be most effective if the most informative terms are processed first. The procedure used in the experiments reported in this paper is as follows:

• Process terms in increasing order of acoustic cost.

• Delay processing terms which trigger more than a certain threshold number of entries. This threshold is determined based on a cumulative histogram of the number of entries associated with all terms in the query. A list of terms that are not processed is maintained in a secondary list $\mathcal{T}$.

• Process all the terms in the query. Entries are pruned based on the cumulative cost after each term is expanded.

• Next, process the list of *delayed* terms in the list $\mathcal{T}$. Entry pruning based on cumulative cost is very effective in this step because active entries at this stage were generated using more informative terms in the first pass through the query terms.

This basic procedure can be refined further by incorporating a term-entropy criteria in determining the order in which to expand the terms. One way of quantifying the information content of terms is the idea of normalized entropy used in information retrieval [7]. A useful criterion is the term-entry weight

$$w_{te} = (1 - H_t)\frac{c_{te}}{\sum_{e \in \mathcal{E}_t} c_{te}} \quad (6)$$

where c_{te} is the count of term t in entry e, $\mathcal{E}_t$ is the set of all indices of terms present in entry t, and H_t is the normalized entropy of terms in the entire database with N entries and is computed using

$$H_t = -\frac{1}{\log(N)} \sum_{e=1}^{N} \frac{c_{te}}{\sum_e c_{te}} \log\left(\frac{c_{te}}{\sum_e c_{te}}\right). \quad (7)$$

Incorporation of this weight $(-\log w_{te})$ in the entry cost calculation further improves pruning efficiency. The dynamic ranges of the acoustic cost and the entropy costs are not comparable and a scale factor, reminiscent of grammar scale in ASR, is empirically determined.

The effectiveness of the pruning strategies is demonstrated on the $4M$ entry task (Table 2). It is clear that delayed pruning is very effective in reducing the number of entries expanded without any impact on the accuracy. In fact, the delayed processing of terms should give exactly the same results as without delayed processing except for any correct entry that gets pruned during the first stage of expansion.

Pruning	baseline	+delayed	+entropy
# Entries Expanded	$920K$	$260K$	$170K$
Sent. Acc. (%)	90.5	90.5	90.4

Table 2. Results demonstrate the effectiveness of pruning on the $4M$ entry task.

4. EXPERIMENTS

Results are presented for indices generated using 3-grams. A 4-gram, unsmoothed phonotactic language model [5] is used in the first pass. This model is trained on the set of phone lattices $\{L_1, L_2, \ldots, L_N\}$ representing the form entries. The size of the grammar, G, as well as the fully-composed network used by the decoder, CLG, for various database sizes are shown in Table 3. For comparison, the G and CLG for a traditional fully-constrained (FC) recognition network is also shown. It is clear that the memory requirements of the first-pass scales well with vocabulary size. It is clear that FC networks increase in size significantly for this case, whereas the phonotactic model size increases only slightly. If the static size of the network alone were the issue, it can be addressed in a number of ways. One could compose the network dynamically at the cost of increased real-time. Other options include maintaining compressed networks (using standard compression) techniques and uncompressing parts of the network on-demand. However, the problem that is more difficult to address is the dynamic memory allocated during decoding. In the case of the $4M$-entry database, the dynamic memory size for the $4M$-entry FC system grows larger than $0.2GB$ for some utterances at large beams. The dynamic memory requirements for the index access stage of vForms is minimal ($< 10MB$) by comparison. The shortlist sizes used in these experiments is only 200 which makes the second-pass recognition trivial.

# form entries	vocab size	phonotactic		FC	
		G	CLG	G	CLG
250K	85K	4.4M	13M	8.4M	41M
4M	400K	9.1M	27M	112M	363M
8M	560K	11.0M	31M	220M	650M

Table 3. Comparison of memory sizes of grammars and recognition networks for phonotactic (4-gram) first-pass recognition and fully-constrained recognizers. K and M stand for 10^3 and 10^6 and interpreted as kilo/mega bytes when referencing memory sizes.

The test database consists of 3668 sentences collected over the telephone in which the users speak the first name, last name, city, and state. The *sentence* accuracy of the various systems are shown in Table 4.

# form entries	FC		vForms	
	sent. acc.(%)	rt factor	sent acc (%)	rt factor
250K	94.6	0.40	95.1	0.44
4M	90.5	0.63	90.4	0.61
8M	85.2	0.75	85.2	0.68

Table 4. Comparison of sentence accuracy and real-time factor for two systems: traditional fully-constrained grammar (FC), and two-pass vForms. Real-time factor is defined as the ratio of the time taken to process the audio to the duration of the audio (small factors indicate faster ASR). Note that the accuray and real-time factors for the two systems are comparable, but the memory requirements for the FC system are significantly greater (Table 3).

It is clear from Table 4 that the vForms system is competitive in accuracy and speed with a traditional system using a fully-constrained grammar, and requires significantly less memory. This result is significant because two-pass recognition schemes have almost never matched the real-time performance of fully-constrained single-pass recognizers. This is because constraints are invoked later in two-pass systems than in one-pass systems. One reason vForms is efficient is because the first-pass recognizer is run with a small beam. If only the top choice phone string is used for short-list generation, this would increase the error rate significantly. Since a phone lattice is used to generate the query, it is only necessary for the correct path to be present in the lattice for the overall system accuracy to remain high. Having demonstrated the advantages of vForms over conventional ASR, the next step is to explore the use of vForms for much larger listings such as state-wide or nation-wide telephone directories where conventional FC systems are infeasible.

A histogram of the the rank of the correct entry in the shortlist is plotted in Fig. 2 for the $4M$-entry database. The correct entry ranks first almost 75% of the time, which shows that index access using n-gram terms is extremely effective. All the rank values of > 100 have been included in the last bin of the histogram. This shows that the correct entry ranks very low in the shortlist about 7% of the time. It is not entirely clear why this is the case. It is possible that some of the terms that appear in those entries are missing in the query. Further analysis is necessary to determine ways of recovering those entries.

5. CONCLUSIONS

The vForms algorithm was shown to be competitive in terms of real-time performance and accuracy with a conventional fully-constrained one-pass system. The new approach scales well with the vocabulary size and the size of the directory, whereas conventional fully-constrained ASR systems become impractical for large directories. Future work will be directed at exploring the use of vForms on much larger applications such as the retrieval of entries from state-wide or nation-wide directory listings.

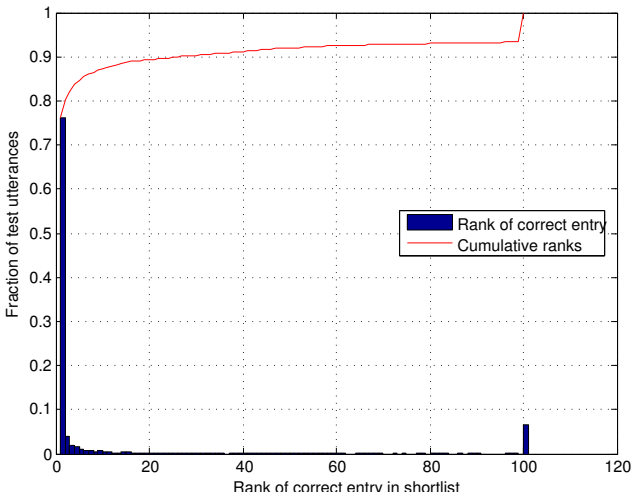

Fig. 2. Histogram of the rank of correct entry in the shortlist.

6. REFERENCES

[1] S. Parthasarathy, C. Allauzen, and R. Munkong, "Robust access to large structured data using voice form-filling," in *Proc. INTERSPEECH*, 2005, pp. 2493–2496.

[2] E. Levin and A. Mane, "Voice user interface design for automated directory assistance," in *Proc. INTERSPEECH*, 2005, pp. 2509–2512.

[3] H. Schramm, B. Rueber, and A. Kellner, "Strategies for name recognition in automatic directory assistance systems," *Speech Commun.*, vol. 31, no. 4, pp. 329–338, 2000.

[4] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer, Speech, and Language*, vol. 16(1), pp. 69–88, 2002.

[5] C. Allauzen, M. Mohri, and B. Roark, "A general weighted grammar library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata (CIAA'2004)*, 2004, pp. 23–34, http://www.research.att.com/sw/tools/grm.

[6] C. Allauzen, M. Mohri, M. Riley, and B. Roark, "A generalized construction of speech recognition transducers," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, 2004, pp. 761–764.

[7] J. R. Bellegarda, "Exploiting latent semantic information in statistical language modeling," *Proc. of the IEEE*, vol. 88, pp. 1279–1296, August 2000.