Robust Access to Large Structured Data Using Voice Form-filling

S. Parthasarathy, Cyril Allauzen

AT&T Labs Research, Florham Park, NJ 07974
{sps|allauzen}@research.att.com

R. Munkong

Georgia Institute of Technology, Atlanta, GA
rmunkong@ece.gatech.edu

Abstract

A method for accurate and scalable *form-filling* by voice is presented. A form consists of a number of fields. Accurate speech recognition is achieved by applying task-specific inter-field constraints. The task constraints are specified typically by providing a database of valid *form-entries*, such as an employee directory containing the name, location, and telephone number. Scalability to very large vocabularies, number of fields, and the ability to accept a variety of user responses, is achieved by a two-pass recognition scheme. An index-based retrieval method is used in the first-pass to produce a shortlist of form-entries. These are rescored in the second-pass to obtain the final result. Experiments on a simple corporate directory access application are presented to demonstrate that the new approach compares favorably, in terms of computing needs, with a traditional one-pass speech recognition system. Experiments on a national street address recognition application are presented to demonstrate that the new approach scales very well to large tasks.

1. Introduction

Many web and telephony applications involve retrieval of information from large structured databases, using *form-filling*. The database typically consists of a number of *fields*. An entry in the database can be retrieved by the user by specifying values for a subset of the fields. In web applications accessed using a computer, entry of fields using a keyboard is simple and accurate. In telephony applications, voice input of fields using automatic speech recognition (ASR) is convenient but error-prone. Every field in the form has to be correctly recognized for the task to be successfully completed. This means that the recognition accuracy for each field has to be very high. An acceptable ASR accuracy can be achieved for simple fields such as account numbers, dates, time, etc., but accurate recognition of names of people or places, airport names, street names, etc. is difficult to achieve if each field is considered individually. However, there are often strong *inter-field* constraints which can be exploited to improve ASR accuracy [1]. 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 methods can get impractical for forms with many fields and large vocabularies. The above discussion applies not only to information retrieval from databases but also to information input. Consider an application in which the user has to schedule a service visit to an address. The address entry form could be designed to produce only valid addresses as provided by, say, the Postal Service.

There are many user interface issues that also have a significant impact on the success of form-filling. The users could specify either the value of one field, or the values of all the relevant fields, in a single utterance. The first option requires that the user select a field either by voice or multi-modal input. In the second option, the ASR system would have to accept a variety of user responses. Finally, there are memory and CPU constraints that impact the design and performance of form-filling systems.

The main goal of this paper is to investigate methods for voice form-filling in scenarios where conventional approaches for applying constraints are not feasible, such as when the sizes of the vocabulary and the database are very large. Another goal is to develop techniques that facilitate automatic creation of form-filling applications.

2. Form-Filling

Form-filling is a commonly used method for gathering user input. Voice input of forms is a difficult problem.

2.1. Voice-based form-filling

Effective form-filling depends critically on the following. **Context-setting**: While a simple form on a display can be used to convey the list of valid fields, a telephone interface with only voice output makes this significantly harder. Careful dialog design is necessary to complete the form successfully. The approach presented in this paper attempts to solve this problem by allowing the user to provide all the necessary information to fill the form in a single, albeit *stylized*, sentence. With the increasing use of phones with displays, especially in VoIP services, the list of fields could be presented visually rather than by voice.
Constraint-modeling: Many applications have strong inter-field constraints. ASR accuracy can be improved significantly by exploiting these constraints. Grammars that represent the constraints exactly may be too large for use in practice. Further, it is not always possible to assume that the fields are spoken in a specified sequence. The approach in Sec. 4 uses a two-pass scheme to apply inter-field constraints.
Field-selection and error-recovery: In speech input-output, it is difficult to designate fields as optional or mandatory. For example, in a customer care application, the easiest way to access the repair status might be to specify an alphanumeric transaction code. However, if the user does not have access to the code, it takes a dialog turn to determine that fact before alternate methods, such as using name or date information, are employed to complete the task. Form-filling would simplify this task by allowing the user to choose the access method. Simple user interfaces with visual output could be designed to allow the user to correct ASR errors in individual fields, greatly simplifying dialog interactions.

2.2. Exploiting task constraints

The main problem addressed in the paper can be stated as follows. Given a database with N_F fields, and a voice utterance (a) that specifies multiple fields in one utterance, (b) in which the order of fields in the utterance is not specified completely, and (c) which may contain limited extraneous filler words, design a high-accuracy ASR system that invokes the inter-field constraints specified implicitly by the database, and is scalable to very large vocabularies and database sizes. Scalability implies that 1-pass, fully constrained speech recognition is infeasible.

3. Traditional approaches

There are a number of obvious approaches to solving the form-filling problem.

3.1. Independent fields

The size of the grammar grows primarily when inter-field constraints are invoked. Two-pass approaches have been shown to be effective in these cases [1]. In the first-pass, the inter-field constraints are ignored and the gramamr, G_I, is assumed to be a concatenation of the grammar for the i^{th} fields given by G_i. Inter-field constraints are invoked in the second-pass to produce the final result

$$r = \mathcal{B}[\Pi_o \left(R_{1-N} \circ G_{1-N}\right)] \tag{1}$$

where, $\circ$ represents composition of transducers, R_{1-N} is the result lattice obtained in the first-pass, G_{1-N} represents the inter-field constraints, and Π_o and $\mathcal{B}$ are the projection to output symbols and bestpath operations, respectively [2].

This approach has a number of disadvantages. Large intermediate lattices need to be generated to prevent empty final results increasing the computational load of the first-pass. As the number of fields increase, the probability that the first-pass result does not contain a single path that satisfies the constraints in G_{1-N} increases. Further, the order of fields may not be known. Multiple ordering of fields increases the grammar size and also the ASR error rate.

3.2. Dynamic grammars

Another commonly used approach for recognition of large directories is to recognize easy fields first to use as constraints for subsequent fields that are more difficult to recognize. For instance, the task of recognizing the names in a telephone directory can be simplified by asking for the city and state first, and constraining the name grammar with that information. This approach still requires significant dialog design to accomodate users that are unable to provide the requested information, such as the city name in this example. Also, real-time generation of dyanmic grammars may be infeasible for large databases.

3.3. Phone recognition and lexical access

One approach to limit the complexity of the first-pass recognition, especially with respect to vocabulary size, is to perform phone recognition. A task specific phonotactic grammar (statistical N-gram grammars) could be used to output a phone lattice in the first-pass, and both lexical and grammatical constraints could be incorporated by rescoring in a second-pass. A serious problem with this approach is that a phone path that satisfies the task constraints may not be present in the first-pass lattice because of inevitable phone insertions and deletions. One solution is to find the path in the first-pass lattice that best matches a path in the constraint grammar to within a weighted edit distance

$$\sum_{r,g} P(r)P(g)d(r,g) \tag{2}$$

where $P(r)$ and $P(g)$ are the probabilities of the paths r and g in the result and constraint lattices respectively, and $d(r,g)$ is an edit distance between the paths r and g [3]. This is computationally expensive for large constraint grammars. One approximation is to use N-best phone strings from the first-pass result and select the string that minimizes the edit distance as the final result.

4. Scalable, effective, two-pass approach

The approaches presented in Sec. 3 are usually effective for tasks with limited complexity (vocabulary size and number of fields) but do not scale well with complexity. A first-pass that consists of phone recognition is a convenient way to design a scalable system that is independent of the size of the vocabulary and the database. The proposal is to use the phone lattice result generated in the first-pass to query a database and generate a shortlist of possible database entries. This is similar to a document retrieval problem in which each entry of the database is treated as a document. There are many existing indexing applications in which a speech database is represented as an indexed phone lattice, which can be queried by text [4]. The form-filling problem is roughly the reverse in the sense that the database is text and the query is done by speech. Once a shortlist is generated, it can be rescored using all available lexical and inter-field constraints to get the final result. Details of this approach are explained in this section.

4.1. Index generation:

An entry in a database is typically a sequence of fields. The first step consists of converting each entry into a phone lattice, L_i. This lattice is designed to incorporate transformations that help maximize coverage of user utterances, such as multiple pronunciations of words, reordering of fields, and multiple ways of speaking entities such as natural numbers. Then, a transducer T_i is constructed that associates with each factor x (diphone, triphone, etc.) that appears in L_i, the database entry index i. The final index

$$\mathcal{I} = \mathsf{Det}_{\mathsf{log}} \left(T_1 \cup T_2 \cup \ldots \cup T_I\right) \tag{3}$$

where $\mathsf{Det}_{\mathsf{log}}$ refers to determinization in the log semiring [4]. For large databases, the index size can grow unmanageably large. In the experiments presented in Sec. 5, there are two ways in which that growth is managed. The first is to restrict the index to factors of interest, say triphones and tetraphones. The second is to split the index into sub-indices.

An example of index creation is shown in Fig. 1. In this example, the vocabulary consists of four units { a, b, c, d } and the list entries in the database are sequences {a b c} and {a b a d} as shown in Fig. 1(a) and Fig. 1(b). The index generated using all factors is shown in Fig. 1(c). A more compact index (Fig. 1(d)). can be generated by restricting the factors to trigrams.

4.2. Generation of phone lattices

The first-pass recognition uses a N-gram phonotactic grammar to produce a phone lattice result. The database (corpus) used to

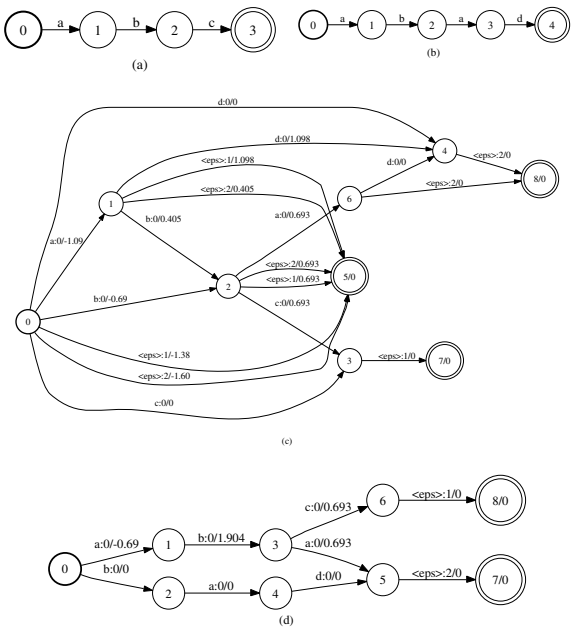

Figure 1: Index creation example. (a) and (b) Two entries in the database. (c) Index transducer mapping all factors in the list entries to the entry index. (d) Index restricted to trigrams.

train the task-specific phonotactic model is often the same collection of phone lattices, $\{L_1, L_2, \ldots, L_I\}$, used to train the index. The N-gram model is trained using conventional language modeling tools [5]. It has been found experimentally that an unsmoothed model, one that allows only those phone N-grams that have been seen in the training data, provides a significant increase in the inclusion rate of the correct index entry in the shortlist.

4.3. Generation of shortlists

Given an index $\mathcal{I}$ and a phone lattice R generated by the first-pass ASR, the short list is generated in the following steps.

1. Any units that are insignificant such as silence and other filler words are removed from R to produce a result R^1.
2. Then the costs in R^1 are normalized so that the best path has a cost of 0.
3. For efficiency, only the factors that are used in the index are retained to produce a cost-normalized, query
$$Q = \mathsf{Det}_{\log}[\Pi_o(R^1 \circ T_F)] \tag{4}$$
where T_F is a transducer that retains only the factors of interest.
4. The list of indices that contain the factors in Q and the associated cost is given by
$$I^1 = \Pi_o(Q \circ \mathcal{I}) \tag{5}$$
The N-best lowest cost indices, computed by $\oplus_{\log}$ over all factors, are given by
$$I^s = \mathcal{B}_{\mathsf{trop}}[\mathsf{Det}_{\log}(I^1)] \tag{6}$$

For very large databases, $\mathcal{I}$ can be represented as a union of sub-indices and the above steps can be performed in parallel for each sub-index.

4.4. Second-pass recognition

Once the shortlist is obtained, the final result can be obtained in a number of ways. The first step is to create a grammar from the shortlist. The index access process is independent (almost) of the order of the fields since it is based on the expected cost of the occurrence of each factor in the query. The dynamic grammar G_2 for use in the second-pass recognition or rescoring is constructed such that it covers common variations in the user utterences such as field transpositions. The size of this grammar is manageable for reasonable size shortlists, say less than 1000. Grammar G_2 can then be used (a) in second-pass decoding, (b) as constraint grammar in recognition based on minimizing the edit-distance, or (c) as grammar for rescoring the first-pass results using *gappy* phone matches to allow for extraneous speech and phone insertions.

In the experiments in Sec. 5, the final result is obtained by a second-pass recognition using grammar G_2.

5. Experiments

Experiments were performed on data collected from two applications. The first is an application to locate an employee in a corporate database by speaking the name. The size of this database is small enough that a conventional one-pass recognition is feasible. In general, the earlier the constraints are invoked the more efficient the decoding. There is no expectation that the new approach will be more efficient in terms of speed or memory for this application. The only goal is to demonstrate that the approach presented in this paper has little overhead even for small databases, even though it was developed to handle very large databases. Of course, an employee locator application can get fairly complicated when one has to resolve multiple listings, etc., and the new approach may be useful in these situations.

The second application involves the recognition of street addresses in a United States Postal Service database. This application was chosen to demonstrate the scalability and effectiveness of the new approach on large databases.

The ASR performance is measured using the *sentence* (or form) accuracy and not word accuracy. Further, the recognizer is set up to output phone strings rather than word strings. The sentence accuracy will not be affected by this decision since the output string is scored as correct if the phone string output by the recognizer is contained in the reference phone lattice for a particular database entry. An advantage of this way of scoring is that homonyms do not need special treatment in the scoring.

5.1. Directory access

The database consists of 138K entries in which each entry consists of the first and last name of an employee. The speech corpus was collected during a trial of an employee locator service over the telephone. This is a challenging speech recognition problem because a significant fraction of names are of foreign origin, and also many of the callers speak English as their second language. The users speak the first and last name in a single utterence.

The ASR performance for this task is presented in Table 1. The baseline system uses a 1-pass grammar which is constrained to produce one of 138K full names. The 2-pass method uses a 4-gram unsmoothed phonotactic model in the first pass to generate phone lattices. A shortlist of 800 entries is generated and rescored to obtain the final result. This baseline represents the best result on this task. The size of the network to

System	Net Size I-pass (Mb)		Sentence	RT
	G	CLG	Acc (%)	Factor
FC 1-pass	2.6	7.5	84	0.23
2-pass	2.4	6.2	83.7	0.24*

Table 1: Comparison of Fully-Constrained (FC) 1-pass recognition and the new 2-pass method on a name recognition task. * RT includes only the first pass.

represent the phone grammar is shown in the grammar column (G). The AT&T recognizer is set up to accept a fully composed and optimized network (CLG) for improved efficiency in decoding [6]. The size of CLG, which is a function of the acoustic model, is also given in the table. The results indicate that the new approach does not incur a significant real-time penalty even though the first-pass does not invoke the lexical constraints. The benefits of this approach will become obvious for more complex queries such as *first name* at *location*, or *last* at *organization*.

The inclusion rate of the correct name in the shortlist of size N is shown in Table 2. An index of 3-grams is used in this experiment. The top choice in the ordered shorlist is the correct name about 67% of the time. The inclusion-rate in the top 800 is over 90%. In this application, there isn't much redundancy across the fields. The size of the shortlist for a given inclusion rate will reduce significantly as the number of fields increase. This is demonstrated in the street address application.

Shortlist size (N)	1	10	100	800
Inclusion rate (%)	67	79	87	92

Table 2: Inclusion rate of the correct index entry in shortlists of various sizes for the names task.

5.2. Street address recognition

The database consists of 35M US postal service street addresses. The task consists of speaking addresses that range from somewhat unusual such as *1 3/4 1 1/2 Avenue Prairie Farm Wisconsin 5 4 7 6 2*, to common such as *10 Mile Road Clarendon North Carolina 2 8 4 3 2*. The size of the vocabulary is about half a million words and there are approximately 5 million unique street addresses (ignoring the house numbers on each street). The speech corpus consists of users speaking a complete street address in a single utterance over the telephone.

System	Net Size I-pass (Mb)		Sentence	RT
	G	CLG	Acc (%)	Factor
FC 1-pass	227	499	90	1.5
2-pass	9.8	20	90	0.3*

Table 3: Comparison of fully-constrained 1-pass recognition and the new 2-pass method on an address recognition task. * RT includes only the first pass.

The parameters of the baseline and 2-pass systems in this experiment are the same as in the directory access task except for the following: (i) the phonotactic model used in the first pass was trained on the address corpus instead of the names corpus, (ii) the shortlist size is reduced to 80 instead of 400. The results are shown in Table 3. It is obvious that the two-pass approach has a significant advantage in this case. If the order of the fields is not constrained (it is in this example), the advantage of the new approach is likely to be even more significant.

Shortlist size (N)	1	10	20	80
Inclusion rate (%)	47	67	77	92

Table 4: Inclusion rate of the correct index entry in shortlists of various sizes for the address task.

The inclusion rate as a function of the size of the shortlist is shown in Table 4. The index-access procedure is much more effective in this case because of the redundancy across fields.

One caveat in interpreting the real-time performance is that the time taken for the index access as well as the second-pass recognition has not been included. However, the second-pass is very efficient and requires only a small fraction of the time taken for the first pass. Index access depends on the size of the first-pass phone lattice which can be kept small enough to make the index access very efficient. The shortlist size is on an average about a hundred and so the second-pass recognition will not affect the total run-time significantly. These issues will be investigated more carefully on more complex tasks currently being implemented.

6. Conclusion

A new two-pass method for form-filling by voice is presented. A phone lattice is produced in the first pass which is used to query an index to obtain a shortlist of valid entries. This shortlist is rescored in the second-pass to obtain the values of the individual fields.

This approach was tested on two applications. The results demonstrate that the new approach does provides flexibility without incurring significant penalty for small tasks, and provides a significant advantage in terms of computational needs for very large tasks.

7. References

[1] R. C. Rose, S. Parthasarathy, B. Gajic, A. E. Rosenberg, and S. Narayanan, "On the implementation of asr algorithms for hand-held wireless mobile devices," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, May 2001, pp. 7–11.

[2] M. Mohri, F. C. N. Pereira, and M. Riley, "The Design Principles of a Weighted Finite-State Transducer Library," *Theoretical Computer Science*, vol. 231, pp. 17–32, January 2000, http://www.research.att.com/sw/tools/fsm.

[3] M. Mohri, "Edit-distance of weighted automata," in *Seventh International Conference on Implementation and Application of Automata*, 2002.

[4] C. Allauzen, M. Mohri, and M. Saraclar, "General indexation of weighted automata - application to spoken utterance retrieval," in *Proc. of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT/NAACL*, 2004, pp. 33–40.

[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] 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.