EXPERIMENTS IN KEYPAD-AIDED SPELLING RECOGNITION

S. Parthasarathy

AT&T Labs Research, Florham Park, NJ 07974
(sps@research.att.com)

ABSTRACT

Accurate recognition of spellings is necessary in many call-center applications. Recognition of spellings over the telephone is inherently a difficult task and achieving very low error rates, using automatic speech recognition, is difficult. Augmenting speech input with input from the telephone keypad can reduce the error rate significantly. A variety of schemes for combining the keypad and speech input are presented. Experiments on a name entry task show that spellings can be recognized nearly perfectly using combined keypad and speech input, especially when a directory lookup is possible.

1. INTRODUCTION

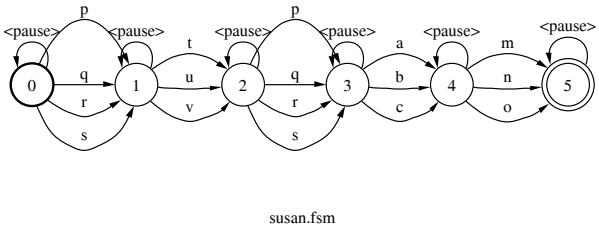
Automatic speech recognition (ASR) systems that are being deployed today have the ability to handle a variety of user input. A typical call-center transaction might begin with a fairly unconstrained natural language statement of the query [1] followed by a system or user initiated input of specific information such as account numbers, names, addresses, etcetera. A transaction is usually considered successful if each of the input items (fields) is correctly recognized, perhaps with repeated input or other forms of confirmation. This implies that each field has to be recognized very accurately for the overall transaction accuracy to be acceptable. In order to achieve the desired accuracy, state-of-the-art ASR systems rely on a variety of domain constraints. For instance, the accuracy with which a 10-digit account is recognized may be 90% using a digit-loop grammar but close to perfect when the grammar is constrained to produce an account number which is in an account-number database. Similarly, if one has access to a names directory and the user speaks a name in the directory, the performance of ASR systems is generally fairly good for reasonable size directories.

In some applications, the use of domain constraints is problematic. As an example, consider an application whose purpose is to enroll new users for a service. In this case, information such as the telephone number, name etc., need to be obtained without the aid of database constraints. One could still use *a priori* constraints, such as a names directory that covers 90% of the US population according to the US Census data, to improve recognition accuracy. However, if the names distribution of the target population does not match the US Census distribution, the out-of-vocabulary (OOV) rate could be substantially higher than 10%.

Recognition of long digit-string, names, spelling, etc. over the telephone, whether human or machine, is inherently difficult [2] [3]. Humans recover from recognition errors through dialog. Such dialogs, which might involve a prompt to repeat a portion of the digit string or a particular letter in a name, have been implemented in ASR systems but with limited success [4] [5]. In the short-term, it appears that the best way to achieve very accurate recognition of difficult vocabularies such as letters and digits is to use to supplement voice with other input modalities such as keypads (touch-tones). The telephone keypad is designed for numeric entry and therefore is a natural backup modality for digit-string entry. However, the keypad is not so convenient for the entry of letter strings (for spelled names, etc.). *Cluster keyboards*, that partition the letters of the alphabet onto subset keys, have been designed to facilitate accurate letter-string entry using keyboards [6]. The letter ambiguity for each key-press in these keyboards, is addressed by hypothesizing words in a dictionary that have the highest probability according to a language model. Such methods are effective, but they require the use of specialized keypads. If one is constrained to use the standard telephone keypad, one possibility is to use speech for disambiguation. A scheme for integrating keypad and speech input has been presented recently [7]. The main focus of this work was to obtain pronunciations of *new* words using side information from keypad input. Similar strategies for the combined use of the telephone keypad as well as voice for very accurate recognition of spellings is the subject of this paper.

2. SPELLING RECOGNITION STRATEGIES

There are a number of ways to improve the performance of spelling recognition using the constraints provided by keypad input. The various options are explored in the following sections.



susan.fsm

Fig. 1. Unweighted grammar generated from the keypad input 78726.

2.1. Recognition constrained by keypad input

There are either 3 or 4 letters of the alphabet that are associated with each key on the telephone keypad. The simplest scheme for using keypad input as well as speech is to first get the keypad sequence and dynamically construct a grammar that permits all letter sequences that map to the given keypad sequence. An example network is shown in Fig. 1.

In this example, the grammar is unweighted. If one had access to spellings that characterize the domain, such as a directory of names for the name recognition task, one could use an N-gram letter model L_N trained on this data, and construct a weighted grammar

$$K_w = K \cap L_N \tag{1}$$

where K is the unweighted keypad grammar, and $\cap$ is the intersection operation. If the corpus on which L_N is estimated is large, an *unsmoothed* N-gram model (only those N-grams that appear in the training corpus are allowed) provides a significant advantage.

2.2. Two-pass schemes

The accuracy of spellings using state-of-the-art ASR systems is reasonably good, especially if good language models (of letter sequences) are available. It would be useful to perform spelling recognition using ASR alone in the first pass, and use keypad input only when the ASR confidence is low. This way, the inconvenience of using keypad entry will be limited to those utterances that are poorly recognized for reasons such as the presence of background noise or unusual accents.

One possibility is to use the letter N-gram model L_N in the first-pass recognition. The output of the recognizer is a letter lattice, R_{L_N}. The user is then asked to input the letter string using the keypad (1 press for each letter in the word) and a keypad constraint grammar K is created. The final result is the letter string

$$r = \mathbf{bestpath}(R_{L_N} \circ I \circ K) \tag{2}$$

where $\circ$ denotes the *composition* of finite-state transducers, and I is a transducer that eliminates silences and other filler words in the recognized output.

2.3. Database lookup

Each of the schemes described in 2.1 and 3.3 could be followed by a lookup in a database (of valid words, names, etc.) to find a valid letter sequence. The resulting letter string

$$r_D = \mathbf{bestpath}(R_{nc} \circ D) \tag{3}$$

where R_{nc} is the word lattice obtained in 2.1 or 3.3 without a database constraint and D is a finite state network that accepts only valid letter strings. Implementing database lookup as a separate step from speech recognition has the following advantages.

- The complexity of the recognizer does not grow with the size of the database/directory.
- The vocabulary (allowed letter strings) as well as domain-dependent language models (such as frequency of requested names) could be updated independent of the recognizer, thereby simplifying service deployment.

Another option is the use of the keypad to constrain only the first N letters. For long names, keying in all the letters may be too burdensome, but keying in only the first few may be considered acceptable. This provides a way to tradeoff accuracy for convenience, and combined with a database lookup can be very effective (3.3).

3. EXPERIMENTS

3.1. Distribution of names

The task is the recognition of spelled names. In applications where a directory is not available, a common solution is to attempt to cover as large a target population as possible, using a directory of names obtained from an independent source such as the Census or the Social Security Administration (in the United States). However, one cannot depend on the distribution of names in the target population matching the distribution of the general population of the country. Table 1 shows the out-of-vocabulary (OOV) rate of names taken from three tasks, an AT&T customer service task (open names), and two corporate directories (containing about 50K unique names), for a range of vocabulari sizes. The Census data indicates that 90K most frequent names cover about 90% of the US population. From Table 1, it is clear that the OOV rates can be significantly higher for a given task. The conclusion is that the vocabulary (grammar) of an ASR system designed to recognize names will need to be very large to keep OOV rates low.

3.2. Lexically constrained recognition

The performance of a state-of-the-art letter string recognizer, on a spelled-names task over the telephone, is shown

	OOV - type (token) %		
Vocabulary	Task1	Task2	Task3
100K	14.7 (18.6)	16.1 (36.7)	17.9 (39.0)
200K	9.0 (11.2)	10.1 (23.7)	11.6 (25.5)
800K	3.5 (4.8)	3.7 (9.0)	4.1 (9.3)
1.6M	2.3 (2.9)	2.7 (6.5)	2.8 (6.5)

Table 1. Out-of-vocabulary rates for test names taken from 3 tasks as a function of the size of a given directory.

Unique names	name acc (%)	letter acc (%)	rt factor
124K	92	98.2	0.08
1.6M	83	95.2	0.27

Table 2. Performance of name recognition using a spelled name grammar.

in Table 2. The grammar is constrained to produce only valid names. The acoustic model was trained discriminatively on an collection of independent databases of letter string utterances collected over the telephone. All the test names were in-grammar. The accuracy of name recognition (letter string accuracy) is fairly good at 92% for a 124K vocabulary and degrades to 83% for a vocabulary of 1.6 million names. An accuracy of 83% for name recognition may be considered acceptable in many applications. However, if the name is just one field in a number of fields that need to be filled to complete a task, it may be necessary to operate at much lower error rates to maintain reasonable task completion rates. Another point to note from Table 2 is that the resource requirements (real-time factor on a Pentium desktop with a 1 GHz processor) increases significantly for large grammars.

3.3. Keypad integrated recognition

There are many systems that allow spelling input using just the keypad. Schemes that attempt disambiguation by finding a match in a dictionary are suitable for limited vocabularies. As the size of the vocabulary grows, directory lookup often does not result in a unique entry. Table 3 shows the results of an experiment where a single key-press is used to enter a letter. A directory containing 124K names maps to about 99K unique key sequences. A given key sequence, corresponding to the spelling of a name, results in a unique name after lookup about 48% of the time. The test set of names is the same as the one used in the recognition experiment above. When the directory lookup results in multiple names that match the key sequence, some other mechanism is required to select a single name (or generate an ordered set). In this experiment, a language model (frequency of

Names	Keys	Lookup	LM Lookup
124K	99K	48%	93% (98.4% WER)
1.6M	1.1M	4%	91% (97.8% WER)

Table 3. Performance of name recognition using keypad input only. Each letter is input using 1 key-press.

names according to the US Census) is used to pick the name with the highest frequency of occurrence amongst the set of retrieved names. Since this names distribution of this test sample matches reasonably well with Census distribution, the accuracy of name recognition increases to 93%. For a directory of 1.6 million names, a name is uniquely retrieved only 4% of the time without a Census language model and 91% when the language model is invoked. The risk, however, is pretty high (accuracy could drop from 91% to 4%) when the language model does not match the test data.

The above discussion gives some characterization of the spelled name entry problem. It is clear that solution based on speech or keypad alone may not be acceptable for applications that require highly accurate name entry, given the current state of speech recognition.

The results of name recognition using keypad input to constrain the recognizer (2.1) are shown in Table 4. The first option is to key in every letter in the name (K-∞) and speak the letters. Even with no lookup, the name can be retrieved with an accuracy of 90% (a letter accuracy of 98.4%). At this point, there are no task constraints built into the system. This accuracy can be improved further by using a task-dependent N-gram model, which in this case was trained on the 1.6 million list of names. It is quite interesting that 98% accuracy can be achieved with a vocabulary of about 1.6 million names. When a directory is used for lookup, name recognition is nearly perfect even for 1.6M name directory.

If only the first 3 letters are entered using the keypad, again one 1 key-press per letter, the accuracy of name recognition with no lookup drops to 66% with no language model and 84% with a 4-gram letter sequence model. Directory lookup improves the accuracy significantly to near perfect recognition. Even the entry of the first letter of the name yields accuracies that are much higher than a fully constrained ASR system (improvement from 84% to 94%) for the 1.6M names directory.

As explained in section 3.3, one could reverse the order of the keypad and speech input. The results are shown in Table 5. An unsmoothed 4-gram model is used in the first pass. The name accuracy is a modest 71%. This improves to 91% with a directory lookup for a directory size of 1.6M. Keypad constraints applied in a second pass significantly improve performance. For the K-∞ case, the accuracy improves to 97%, roughly matching the accuracy of the system where speech input follows keypad input. The other num-

	Error rate - name (letter) %		
System	no lookup	124K lookup	1.6M lookup
K-∞	90 (98.4)	100 (100)	100 (100)
K-∞-4grm-uns	98 (99.7)	100 (100)	99 (99.8)
K-3	66 (93.3)	100 (100)	98 (99.5)
K-3-4g-u	84 (96.6)	99 (99.8)	97 (99.4)
K-1	56 (88.9)	97 (99.2)	94 (98.6)
K-1-4g-u	76 (93.4)	94 (98.2)	93 (97.8)

Table 4. Performance of name recognition using combined keypad and speech input. K-∞ implies that the letter string for the complete name is entered using the keypad. K-N implies that only the first N letters are entered using the keypad. 4g-uns means an unsmoothed 4-gram model of the letter sequences. Real-time factor (RTF) for K-∞ condition is 0.01. As constraints are relaxed, the recognizer becomes less efficient, and RTF increases to 0.07 for the K-1 condition.

	Error rate - name (letter) %	
System	no lookup	1.6M lookup
4g-u	71 (92.3)	91 (97.6)
4g-u + K-∞	97 (99.5)	99 (99.8)
4g-u + K-3	84 (96.8)	97 (99.4)
4g-u + K-1	75 (94.2)	93 (97.8)

Table 5. Performance of name recognition using speech input first, followed by keypad entry. The real-time factor for this scheme is in the range 0.1 - 0.4 because the first-pass recognition is not constrained by keypad input.

bers in Table 5 show that the order of speech and keypad input does not really matter and that the performance in either case is very good.

4. DISCUSSION

Recognition of spellings is a challenge for ASR systems as well as humans. The strategies that human listeners employ for spelling recognition and error corrections are very interactive and involve prompts for partial strings, disambiguation using familiar words (S as in Sam), etc. which are not easily implemented in current ASR systems or are not very effective with current technology. Keypad input may not be very natural in a spoken language system and the design of a user interface to incorporate keypad and speech may be a challenge. However, the experiments in this paper have demonstrated that keypad combined with speech can be extremely effective. A variety of schemes were presented for combining speech and keypad input and these provide mechanisms for a tradeoff between accuracy and convenience.

5. CONCLUSION

An effective method of entering spellings over the telephone is presented, that augments speech input with keypad input. A variety of different mechanisms for integrating the two modalities were presented and evaluated on a names task. The results show that letter strings can be recognized very accurately even without directory-based retrieval. When a directory is used for retrieval, name recognition is nearly perfect even for large directories containing up to 1.6 million names.

6. REFERENCES

[1] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may i help you?" *Speech Communication*, vol. 23, pp. 113–127, 1997.

[2] J. C. Junqua, S. Valente, D. Fohr, and J. F. Mari, "An n-best strategy, dynamic grammars and selectively trained neural networks for real-time recognition of continuously spelled names over the telephone," in *Proceedings International Conference on Acoustics, Speech, and Signal Processing (ICASSP 95)*, May 1995, pp. 9–12.

[3] F. Thiele, B. Rueber, and D. Klakow, "Long range language models for free spelling recognition," in *Proceedings International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, 2000, pp. 1715–1718.

[4] M. Orlandi, C. Culy, and H. Franco, "Using dialog corrections to improve speech recognition," in *ISCA Workshop on Error Handling in Spoken Dialogue Systems*, August 2003.

[5] M. Swerts, J. Hirschberg, and D. Litman, "Corrections in spoken dialog systems," in *Proceedings of International Conference on Spoken Language Processing*, October 2000, pp. 16–20.

[6] N. Klarlund and M. Riley, "Word n-grams for cluster keyboards," in *Proceedings of the European Chapter of the Association for Computational Linguistics*, April 2003.

[7] G. Chung and S. Seneff, "Integrating speech with keypad input for automatic entry of spelling and pronunciation of new words," in *Proceedings of International Conference on Spoken Language Processing*, September 2002, pp. 2061–2064.