International Conference on Speech and Language Processing (ICSLP '96),
Philadelphia, USA, October 3-6, 1996

RECOGNITION OF SPELLED NAMES OVER THE TELEPHONE

Hermann Hild and Alex Waibel

hhild@ira.uka.de,ahw@cs.cmu.edu
Interactive Systems Laboratories
University of Karlsruhe — 76128 Karlsruhe, Germany
Carnegie Mellon University — Pittsburgh, USA

ABSTRACT

Recognition of spelled names over the telephone line is essential for applications such as telephone directory assistance, or automatic mail ordering. We present recognition results on the spelling section of the OGI Spelled and Spoken Word Telephone Corpus, using a Multi-State Time Delay Neural Network (MS-TDNN). Many applications allow for strong language modeling constraints. In our experiments we examined the beneficial effects of reducing the search space to a list of last names, ranging from about 1000 to 14 million entries. We compare tree search methods and show that significant improvements can be achieved by enriching the search trees with probabilities.

1. INTRODUCTION

This paper presents recognition results on the OGI Spelled and Spoken Word Telephone Corpus, using a Multi-State Time Delay Neural Network (MS-TDNN). While it is desirable to recognize spelled strings which are embedded in spontaneous speech [1], ("Smith please, thats S-M-I-T-H"), the task here is to recognize letters only. Spelled letters over the telephone are easily confused. Current systems achieve in the order of 90% letter accuracy, which results in string (i.e. name) accuracies far below practical usefullness. Fortunately, many applications, most prominently telephone directory assistance, allow for strong language modeling constraints. We show how very large, but conceptionally simple tree structured finite state grammars (FSG) can achieve very good recognition results ranging from 89 to 98% name accuracy for lists of 1000 up to 14 million last names.

2. THE LETTER RECOGNIZER

A connectionist recognizer, the Multi-State Time Delay Neural Network (MS-TDNN) [3, 5] is used for connected spelled letter recognition. The MS-TDNN integrates the time-shift invariant architecture of a TDNN and a nonlinear time alignment procedure (DTW) into a word-level classifier. The front-end TDNN uses sliding windows with time-delayed connections to compute a score for each phoneme-like state in every frame. Each word to be recognized is modeled by a sequence of phonemes; in a dynamic time warping procedure, an optimal alignment path is found for each word. The activations along these paths are then collected in the word output units. The error derivatives are backpropagated from the word units through the alignment path and the front-end TDNN. For continuous recognition using no or n-gram language models, the standard "one stage dynamic programming search" is used. Algorithms for search in finite state grammars are described below. Training starts with a bootstrapping phase, which involves only the first three layers of the net and establishes a phoneme classification. Then the system is trained on "letter level", using the "classification figure of merit" (CFM) [4] error function for discriminative training.

3. LANGUAGE MODELS

Let $S = \{s_1, \ldots, s_N\}$ be a set of names or strings. In the following we examine techniques which confine the recognition to the names in S. The advantage is a high increase in recognition accuracy. The drawback is that names not in S can not be recognized. In [2], a score (interpreted as probability) is computed for each letter. These scores are used in a tree search to retrieve names from a set of 50,000 names. [1] compares several techniques, which use the constraints either within the search or in a postprocessing step such as nearest neighbor search. In [6], a complex procedure using a mixture of techniques and several recognition passes is proposed. The approach presented below uses very large, but conceptionally simple finite state grammars (FSG) to exactly represent the names in S.

3.1. Search in Finite State Graphs

Each word to be recognized is represented by one acoustic model. Conventional search techniques handle sequences of words by concatenating the models as shown to the left in figure 1. In a FSG, word (in our case letter) sequences can be explicitly coded in a graph structure. Although the same letters use the same acoustic modeling, it is necessary to keep an individual copy for each node in the tree, since it represents a different search history. Each node consumes memory

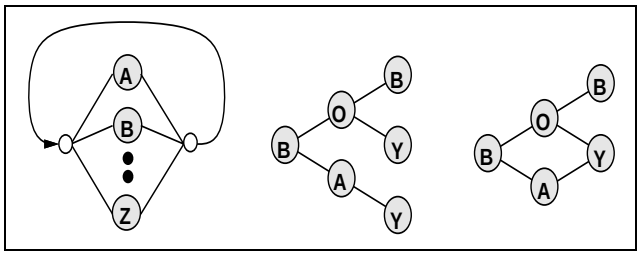

Figure 1: Conventional search (left) uses one instance of each word model. Search in finite state grammars needs multiple copies, as examplified for a tree (middle) and minimal graph (right).

and computing resources, therefore, the FSG should be kept as small as possible, which can be achieved by constructing a minimal graph. Although a tree can be several times as large as a minimal graph, it features one major advantage: Since each represented string s_i is uniquely determined by its final node, no backpointers are needed to identify the best letter sequence at the end of the search.

The largest tree we are employing uses almost 2 million nodes to represent about 800,000 names, which makes a beam search strategy indispensable. A simple scheme is used in the time synchronous search: If s^* has been the highest observed score, all nodes with scores $s < s^* - beam$ are deactivated. Depending on the beam size and the position in the tree, about 50 - 1000 nodes are active at each frame in time. The fact that no backpointers are needed keeps the search tree very simple. Essentially, in each node only one cell for each state of the corresponding acoustic model is needed to store the accumulated search score. Scores are forwarded within a node and to successor nodes, as illustrated in figure 2.

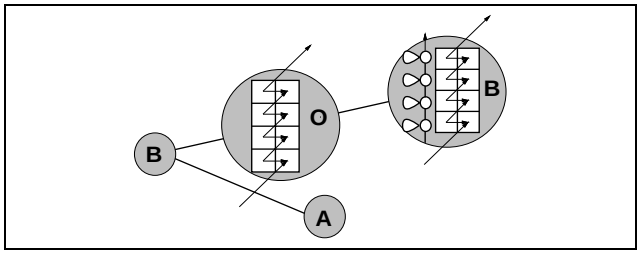

Figure 2: Tree search: Acoustic scores are forwarded within nodes and across nodes.

3.2. Search Tree and Probabilities

Confining the search to a given set S is a strong constraint, but there is one other source of information yet unconsidered. Common names such as "Smith" are very frequent, others rare. By counting their relative frequency, a probability $p(s_i)$ can be assigned to each name $s_i \in S$. $P(s_i)$ is most naturally incorporated into the tree by associating each final node representing s_i with $p(s_i)$. Theoretically, it should not matter where the probability mass is distributed in the tree, as long as the probability along a path to string s_i accumulates to $p(s_i)$. However, as the language model (LM) knowledge becomes only available at the very end of the search, the beam search may cut off eventually good candidates too early. An example for a probability assignment of this and two other methods described below is illustrated in figure 3.

3.3. Local Probabilities

Let the string $s_i \in S$ consist of n_i letters

$$s_i = l_{i_1} l_{i_2} \dots l_{i_{n_i}}$$

A partial path $l_{i_1} l_{i_2} \dots l_{i_k}$ uniquely defines a node in the tree. We denote the unique transition into this node as $t_{i,k}$, i.e.

$$t_{i,k} \equiv l_{i_1} l_{i_2} \dots l_{i_{k-1}} \xrightarrow{t_{i,k}} l_{i_k}$$

Instead of assigning $p(s_i)$ to a final node, we can involve the LM earlier in the search process by defining a "local" probability $\alpha_{local}(t)$ for each transition t:

$$\alpha_{local}(t_{i,k}) := p(l_{i_k} | l_{i_1} l_{i_2} \dots l_{i_{k-1}})$$

$\alpha_{local}(t_{i,k})$ can be computed as the relative frequency by which a path is extended from its parent node into $t_{i,k}$ as opposed to its sibling transitions. We note that the probabilities along the path to the final node representing s_i correctly accumulate to $p(s_i)$:

$$\begin{aligned} \prod_{k=1}^{i_{n_i}} \alpha_{local}(t_{i,k}) &= p(l_{i_1}) * p(l_{i_2}|l_{i_1}) \cdots * p(l_{i_{n_i}}|l_{i_1}..l_{i_{n_i-1}}) \\ &= p(l_{i_1} l_{i_2} \dots l_{i_{n_i}}) = p(s_i) \end{aligned}$$

3.4. Early Probabilities

The most likely string (i.e. final node) that can be reached from a partial path can serve as a measure for the potential "importance" of the path. We define $\beta(t)$ as the highest probability $p(s_i)$ which can be reached from a transition t. $\beta(.)$ can be computed by propagating the maxima from the leaves to the root of the tree:

$$\beta(t_{i,k}) := \begin{cases} p(s_i) & k = n_i \text{ ("final node")} \\ max_{t \in suc(t_{i,k})}\{\beta(t)\} & else \end{cases}$$

Now starting from the root, the transition probabilities are defined as the "missing probability" towards $p(s_i)$:

$$\alpha_{early}(t_{i,k}) := \begin{cases} \beta(t_{i,1}) & k = 1 \\ \frac{\beta(t_{i,k})}{\beta(t_{i,k-1})} & k > 1 \end{cases}$$

Again we note that the final probabilities are correct:

$$\begin{aligned} \prod_{k=1}^{i_{n_i}} \alpha_{early}(t_{i,k}) &= \beta(t_{i,1}) * \frac{\beta(t_{i,2})}{\beta(t_{i,1})} * \cdots * \frac{\beta(t_{i,n_i})}{\beta(t_{i,n_i-1})} \\ &= \beta(t_{i,n_i}) = p(s_i) \end{aligned}$$

Figure 3 shows an example for each of the three methods, which in the experiments to be described will be refered to as "final", "local" and "early". So far we have ignored that a string s_i can be a prefix of another string s_j, which invalidates some of the above formulas. The problem can be fixed by using an explicit "end-of-string" marker $l_{i_{n i+1}}$ for each string s_i.

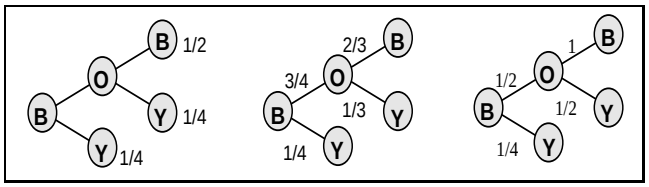

Figure 3: Final (left), local (middle) and early (right) assignment of probabilities for a tree representing the names (Bob, Boy, By) with the probabilities $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

4. EXPERIMENTAL RESULTS

4.1. Data Base

The "Oregon Graduate Institute (OGI) Spelled and Spoken Word Telephone Corpus" provides recordings from about 4000 calls over the public telephone line. Among other prompts ("What city are you calling from?", "what is your last name" etc.), callers were asked to spell their last names with and without short pauses between letters (SLP/SLN), their first names with pauses (SFP), and the alphabet (ALP). About 8% of the spellings contain out-of-alphabet words, e.g. "c h e [sorry] c h a v [as in victor] e z". As our letter recognizer can not yet handle such cases, they were excluded[1] from the experiments. With the partition into training, development and test set provided with the data base, the data used for our experiments amounted to the numbers listed in table 1.

Set	Strings	Letters
Training (SLN, SLP, SFP, ALP)	4132	39687
Dev. Test (SLN, SLP, SFP, ALP)	2063	15612
Test 1 (SLN)	685	4419
Test 2 (SLP)	305	1935

Table 1: Sizes of Training, Crossvalidation and Test Sets

4.2. The Name Lists

A set of lists ranging from 1000 to 14 million names was used to evaluate the tree search procedure. To ensure that all names in the test set are represented[2], we created the lists by filling up the SLN/SLP sets with randomly (without replacement) selected entries from a list of 14 million entries, which was obtained from directory listings from the northeast of the United States. Of course the lists contain many double names. The sizes of all lists, the number of unique entries and the perplexity of the SLN test set given a tree with and without probabilties is shown in table 2.

Total	unique	PP	PP(probs)
685	596	3.28	2.65
1,000	870	3.52	2.79
2,500	1,990	4.17	3.13
5,000	4,078	4.50	3.39
10,000	7,445	5.20	3.66
25,000	15,571	6.11	3.97
50,000	26,787	6.89	4.20
100,000	44,714	7.71	4.40
250,000	85,258	8.93	4.61
500,000	135,550	9.93	4.75
1,000,000	209,301	10.94	4.87
2,000,000	313,320	11.97	4.97
4,000,000	452,903	12.93	5.06
8,000,000	630,718	13.83	5.13
14,000,000	807,013	14.53	5.17

Table 2: Sizes of the names lists, and perplexity on the SLN test set using plain trees and trees with probabilities.

4.3. Baseline Results

16 Melscale FFT coefficients are computed every 10 msec. The MS-TDNN uses a hidden layer with 100 units, which corresponds to a total of only about 34000 parameters (weights). Minimum phoneme duration constraints are computed from statistics on the training data and encoded in the acoustic letter models. The baseline recognition results using no LM, bi- and trigrams are shown in table 3.

4.4. Tree Search

A recognition test with a tree constructed from a list of 1 million (209,301 unique) names was performed to compare the three different methods of assigning probabilities (final, local, early). Figure 4 demonstrates that for small beam sizes, assigning "local" probabilities achieves the best results. As expected, with increasing beam size all three methods perform equally well, but the recognition becomes more time expensive. Figure 5 shows the recognition results on the SLN and SLP sets for lists of various sizes. The usage of probabilities in the tree comes with an astonishing perplexity reduction (see table 2), which is also reflected in significantly better recognition results.

Language Model	LA	SA
SLP, No LM	**90.6**	60.0
SLN, No LM	**88.2**	53.7
SLN, Bigrams	91.0	62.8
SLN, Trigrams	92.5	70.2

Table 3: Baseline results using no LM, bi- and trigrams.

[1] together with some very infrequent cut-offs. Utterances contaminated with any of the 6 transcribed noise classes are used in the training and test sets.

[2] Over 40% of the 800,000 unique names in the 14 million list occur only once, 49 Names of the SLN test set are not in the list!

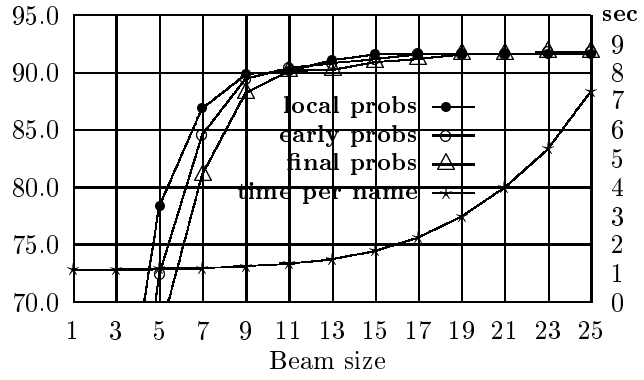

Figure 4: String accuracy for a tree search in a list of 1 million (200,000 unique) names. Three different methods for assigning probabilities in the tree are compared for different beam sizes. The lower curve is the recognition time in seconds for one string.

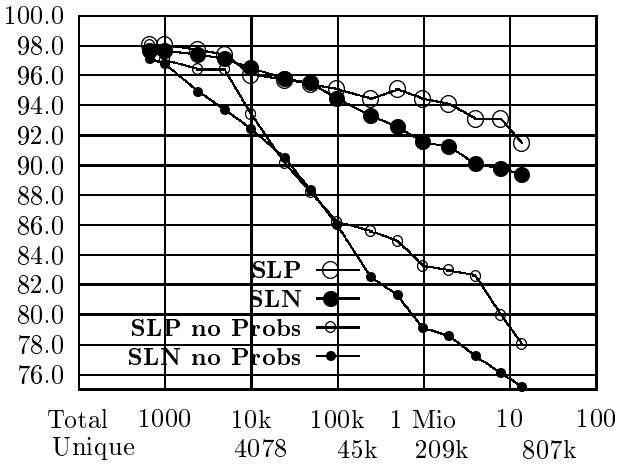

Figure 5: String Accuracy using trees with and without probabilities on the SLN and SLP test sets.

5. SUMMARY

We have tested the MS-TDNN letter recognizer on the spelled names provided in the OGI Spelled and Spoken Word Telephone Corpus. Without using any language modeling, the baseline result on SLP test set is 90.6% LA. As expected the SLN set is more difficult, 88.2% LA was achieved. These results are only about 2 - 3% worse than our results on high quality speech spellings [5, 1] probably because people tend to spell more careful under adverse telephone condition. Bi- and trigrams achieve only moderate improvements. If the search is constrained to a given set of names, high letter and string accuracies can be achieved. Enriching the search tree with probabilities proved to be astonishingly helpful. In lists in the order of 1000 names well above 95% string accuracy can be reached. Even in our largest list with 14 million (800,000 unique) names, almost 90% string accuracy is achieved. The results are summarized in table 4.

Language Model	SLN	
	LA	SA
No LM	**88.2**	53.7
Bigrams	91.0	62.8
Trigrams	92.5	70.2
Tree 1,000	98.1	**97.7**
Tree 100,000	97.5	**94.4**
Tree 1,000,000	97.1	**91.5**
Tree 14,000,000	96.5	**89.3**
" 14,000,000, no prob.	91.4	**75.2**

Table 4: Letter and string accuracies for the SLN and SLP test set, given no language model, bi- and trigrams and trees enriched with probabilities.

Acknowledgements

The authors would like to thank the members of the Interactive System Laboratories, especially Klaus Ries and Bernhard Suhm for their assistance with the bi- and trigram language models, and Michael Finke for contributing the idea of the "early" tree probabilities.

6. REFERENCES

1. Martin Betz and Hermann Hild. Language Models for a Spelled Letter Recognizer. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 856–859. IEEE, May 9-12 1995.
2. Roland A. Cole, Mark Fanty, Gopalakrishnan, and Rik D.T. Janssen. Speaker-Independent Name Retrival from Spellings using a Database of 50,000 Names. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Toronto, Ontario, Canada, May 1991. IEEE.
3. P. Haffner, M. Franzini, and A. Waibel. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In *Proc. International Conference on Acoustics, Speech, and Signal Processing.* IEEE, May 1991.
4. J. Hampshire and A. Waibel. A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. In *Proceedings of the 1989 International Joint Conference on Neural Networks*, June 1989.
5. Hermann Hild and Alex Waibel. Speaker-Independent Connected Letter Recognition With a Multi-State Time Delay Neural Network. In *3rd European Conference on Speech, Communication and Technology (EUROSPEECH) 93*, pages 1481 – 1484, September 1993.
6. Jean-Claude Junqua, Stephane Valente, Dominique Fohr, and Jean-Francois Mari. An N-Best Strategy, Dynamic Grammars and Selectively Trained Neural Networks for Real-Time Recognition of Continuously Spelled Names over the Telephone. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 852–855. IEEE, May 9-12 1995.