To appear in: *Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing*, Detroit, USA, May 1995.

LANGUAGE MODELS FOR A SPELLED LETTER RECOGNIZER

Martin Betz and Hermann Hild

Interactive Systems Laboratories
University of Karlsruhe — 76128 Karlsruhe, Germany
Carnegie Mellon University — Pittsburgh, USA

ABSTRACT

In some speech recognition applications, it is reasonable to constrain the search space of a speech recognizer to a large but finite set of sentences. We demonstrate the problem on a spelling task, where the recognition of continuously spelled last names is constrained to 110,000 entries (= 43,000 unique names) of a telephone book. Several techniques to address this problem are compared: recognition without any language model, bigrams, functions to map a hypothesis onto a legal string, n-best lists, and finally a newly developed method which integrates all constraints directly into the search process within reasonable memory and time bounds. The baseline result of 56% string accuracy is improved to 62, 85, 88, and 92%, respectively.

1. INTRODUCTION

Spelled letter recognition is an essential subtask of many speech recognition systems. Applications include spelling of arbitrary sequences (e.g. "license plates"), "repair" or "new word" dialogues for interactive recognizers, and spelling of names or addresses. In the latter two categories the search space can be constrained to a large dictionary of words or names. Constraints can become effective *within the search process* as n-grams or in a fully constrained search. They also can be used to *postprocess* the recognized hypotheses by mapping them onto legal strings, or by finding the highest ranking legal hypothesis in an n-best list. In this paper, we will demonstrate our letter recognizer and the effects of various language models and search techniques on the task of spelled name recognition. Related work on isolated letters was reported by Cole et. al.[2].

2. THE LETTER RECOGNIZER

The Multi-State Time Delay Neural Network (MS-TDNN) [3, 5] integrates the time-shift invariant architecture of a TDNN and a nonlinear time alignment procedure (DTW) into a high accuracy word-level classifier. Figure 1 shows an MS-TDNN in the process of recognizing the excerpted word '`B`', represented by 16 melscale FFT coefficients at a 10-msec frame rate. The first three layers constitute a standard TDNN, which uses sliding windows with time-delayed connections to compute a score for each phoneme-like state in every frame. These scores are the activations in the "Phoneme Layer". In the "DTW Layer", each word to be recognized is modeled by a sequence of phonemes, and 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 back-propagated from the word units through the alignment path and the front-end TDNN.

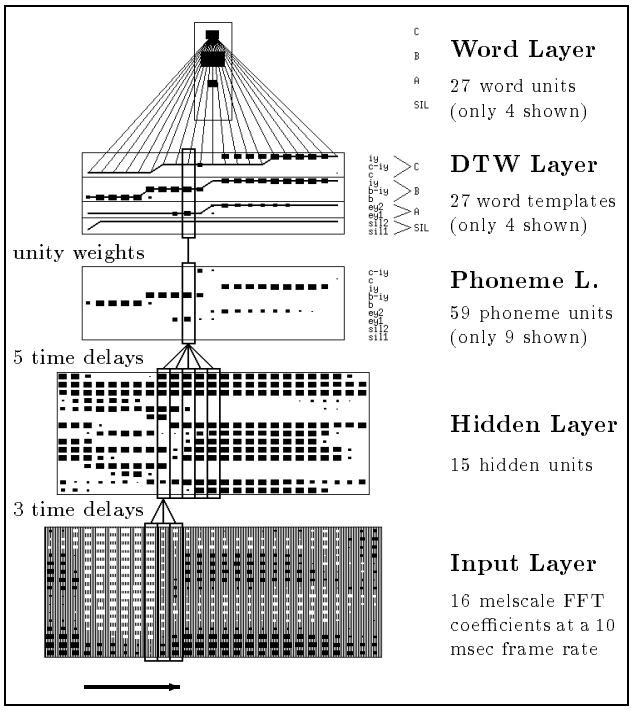

Figure 1: The MS-TDNN recognizing the excerpted word '`B`'. Only the activations for '`SIL`', '`A`', '`B`', and '`C`' are shown.

3. EXPERIMENT SETUP

The "Telephone Directory" used to constrain the search space contained 111,882 entries, with a total of 32,267 unique last names. After accounting for multiple pronunciation alternatives of some letters, the final list of names contained 43,181 strings, referred to as the string set $S = \{s_1, s_2, ...\}$. The recognizer was trained with 8,133 strings (55,449 letters) spelled by 70 speakers. The test set consists of 1,316 strings $\in S$ (8,661 letters) spelled by 23 additional speakers. Speech data were sampled at 16 kHz with a Sennheiser close-talking microphone. Except for the language modeling, the same test setup was used for all experiments.

4. BASELINE RESULTS

As a baseline experiment, the recognizer was tested without any language model, i.e. any letter can follow any other letter with the same probability. The results improve only slightly if bigrams are used, as shown in table 1.

	string acc. (%)	word acc. (%)
no LM	56.4	90.1
bigrams	62.1	92.0

Table 1: Results with no and weak language models.

5. POSTPROCESS SEARCH RESULTS

If the plain recognizer returns a hypothesized name h which is "illegal", i.e. $h \notin S$, we can either try to find a "best match" in S, or ask the recognizer to provide more hypotheses, hoping that one of them will match.

5.1. Closest Match

If h is the hypothesized string returned by the recognizer, we are interested in the name s^* which is "closest" to the recognized string, i.e.

$$s^* = s_{i^*}, \quad i^* = \text{argmin}_i\{d(h, s_i)\}$$

where $d(h, s_i)$ is a distance measure between strings h and s_i. A reasonable choice for $d(h, s_i)$ is simply to count the minimum number of insertions, deletions and substitutions necessary to convert h into s_i, which can be found by an DTW alignment procedure. In addition, one can exploit the fact that some letters are more easily confused than others. For example, `B` is closer to `D` than to `X`, therefore $d(\texttt{B}, \texttt{D})$ should be smaller than $d(\texttt{B}, \texttt{X})$. Instead of simply using penalties of 0 and 1, we define

$$pen(w_i|w_j) = 1.0 - p(w_i|w_j)$$

where $p(w_i|w_j)$ is the probability (estimated from confusion statistics on the training data) that the correct word is w_i given that w_j was recognized. As shown in table 2, this simple measure is quite effective.

closest match	string acc. (%)	word acc. (%)
0 / 1 penalties	75.6	92.8
$1 - p(w_i\|w_j)$ pen.	85.0	95.2

Table 2: Results for mapping the hypotheses to the best matching entry in the string list.

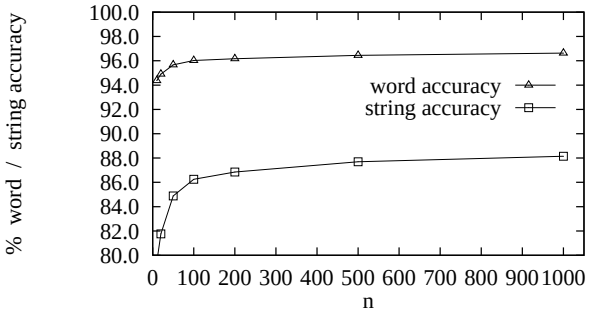

Figure 2: String and word accuracy as a function of the size of the n-best list.

5.2. N-Best

If an n-best list is used, the recognizer returns a (best-to-worst) sorted list $H = (h_1, h_2, \ldots h_n)$ of hypotheses. From this list the best hypothesis h^* matching an entry in the string list S is selected (or the first-best if no match):

$$h^* = \begin{cases} h_1 & \text{if } H \cap S = \emptyset \\ h_{i^*}, \quad i^* = min\{i|h_i \in S\} & \text{otherwise} \end{cases}$$

The recognition accuracy increases with the size of the n-best list. For $n = 50$, the string accuracy is 85%. Saturation occurs at approx. $n = 500$ at 88% string accuracy, as shown in figure 2. In most cases (60.7%), the first-best choice matches an entry in the list. About 5% of these first-best choices are incorrect. In 5.1% of all cases, none of the n hypotheses has a match in the dictionary. As expected, the percentage of misrecognitions increases as the first match occurs further down the n-best list. More detailed statistics are shown in table 3.

position	%	correct	incorrect
1	60.7	763	36
2	10.5	130	8
3	5.1	54	13
4	2.4	28	4
5	1.2	15	1
6	1.1	14	1
7	1.3	11	6
8	1.0	11	2
9	0.8	9	1
10	1.1	10	5
11 - 20	3.6	35	13
21 - 30	2.1	19	8
31 - 40	1.3	11	6
41 - 50	0.7	7	2
51 - 60	0.7	7	2
61 - 70	0.5	5	2
71 - 80	0.5	4	3
81 - 90	0.1	1	0
91 - 100	0.2	1	1
none	5.1	0	67

Table 3: The histogram shows with which frequencies the best matching hypothesis was found at various positions in the n-best list.

6. FULLY CONSTRAINED SEARCH

In the previous section, constraints were applied *after* the hypotheses were already generated; in this section we propose a method to integrate all constraints directly into the search process. For that purpose a finite state grammar (FSG) is constructed which represents exactly the strings in S. The naive approach is to use an FSG with one transition for every letter, which results in 345,570 transitions. A more economic solution is to represent S as a tree (141,066 transitions), or to construct a minimal graph (`minFSG`, 57,713 transitions) (Figure 3).

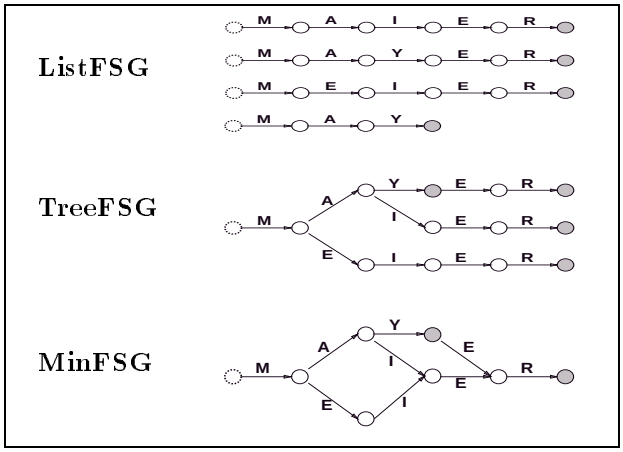

Figure 3: Different types of graphs to represent the strings **MAYER, MAIER, MEIER, MAY**.

Every transition in the graph represents a word in a different context. For example, in the `minFSG` graph, the letter `E` occurs in over 5,800 transitions. Since the full left context of a string is considered during the search, each transition may have a different individual accumulated search score. Therefore, if the conventional one-stage dynamic time warping (DTW) search[6] is employed, one individual word model is needed in the DTW search matrix for every transition in the `minFSG` graph. With a total of 57,713 transitions, this results in a prohibitively time and memory consuming search process. To remedy the problem, we use a technique similar to the "Two-Level DP-Matching" [7]: The letter `E` occurs in over 5,800 different contexts, but the partial optimal score *through* its word model is independent of the search context. This can be exploited by precomputing optimal partial score $s(w_k, t_i, t_e)$ for all words w_k, where t_i and t_e are the points in time where the path enters and leaves the word. The search can also be viewed as a graph search problem through the `minFSG` graph, as explained in figure 4.

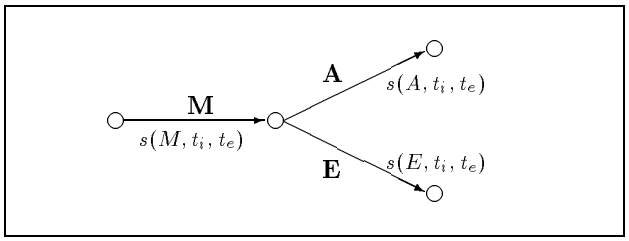

Figure 4: The fully constrained search viewed as a graph search problem. The goal is to find an optimal path through the graph. Each transition corresponds to one letter. The partial score of a transition depends on t_i and t_e.

Figure 5 illustrates some implementational details. For each frame, the search keeps a list of k partial hypotheses, in which the position in the `minFSG`(the search history), the accumulated score, and some other housekeeping data are stored. At time t, each hypothesis is expanded to all allowed successor words w_{suc}, which means that all hypotheses at frames

$$t_e = \{t + mindur(w_{suc}), \ \ldots \ t + maxdur(w_{suc})\}$$

are updated. Good results are achieved with $k >= 40$ active hypotheses. Significant loss in performance was observed with $k < 20$ hypotheses. With beam search, the "Forward-Backward Algorithm" [1] and some other pruning techniques, the search time could be reduced from 12 sec/string to about 3 secs/string (on a HP 735 workstation), still achieving 92.7% string and 97.7% word accuracy. However, this is still almost one order of magnitude slower than the recognition without any language model.

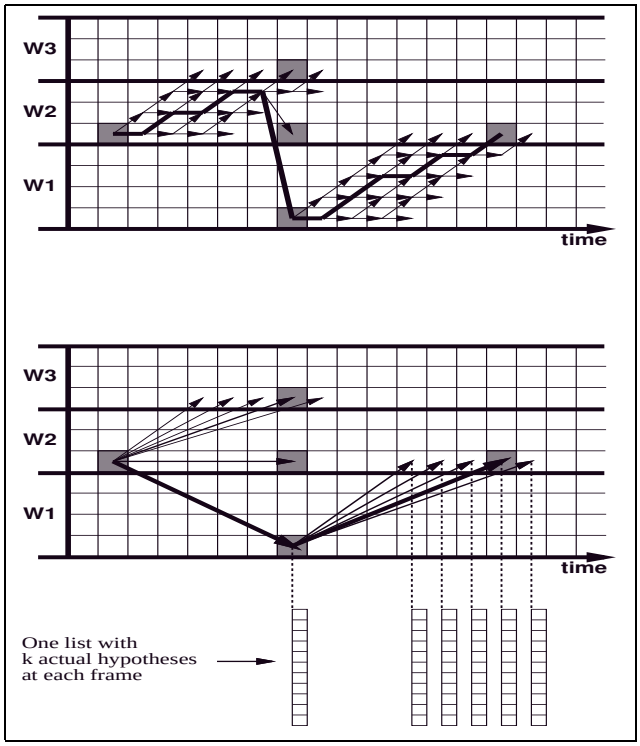

Figure 5: **Top:** Conventional DTW search technique. The matrix contains the prohibitive amount of 57,713 word models, one for each letter in the `minFSG`. **Bottom:** Two level search with only one word model for each letter in the alphabet, but an expensive precomputation of partial scores. At each frame, a list with the active search hypotheses is kept.

7. CONCLUSIONS

In this paper we compared several techniques to constrain speech recognition to a given large list of strings. The task was to recognize spelled letters from a telephone book with 43,000 unique last names.
Bigrams provide only weak constraints. The "n-best" and "closest-match" techniques postprocess hypotheses that were found by the plain recognizer without language model constraints. The best results were achieved with a newly developed technique which allows the integration of all constraints directly into the search process. Table 4 summarizes the results.

	string acc. (%)
No language model	56.4
Bigrams	62.1
Closest match	85.0
N-best	88.1
Full constraints	92.7

Table 4: Summary of results for different language models and search techniques.

Acknowledgements

This research was partly funded by SIEMENS AG and by grant 413-4001-01IV101S3 from the German Ministry of Science and Technology (BMFT) as a part of the VERBMOBIL project. The authors would like to thank all members of the Interactive Systems Labs who contributed with discussions and support, especially Monika Woszczyna for helping out with the *n*-best search module from the JANUS[8] speech recognizer, and Alex Waibel.

8. REFERENCES

[1] S. Austin, R. Schwartz, and P. Placeway. The Forward-Backward Search Algorithm. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 1991.

[2] R. A. Cole, M. Fanty, Gopalakrishnan, and R. D. Janssen. Speaker-Independent Name Retrival from Spellings using a Database of 50,000 Names. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing*, Toronto, IEEE, May 1991.

[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] H. Hild and A. Waibel. Multi-Speaker/Speaker-Independent Architectures for the Multi-State Time Delay Neural Network. In *Proc. International Conf. on Acoustics, Speech, and Signal Processing*. IEEE, 1993.

[5] H. Hild and A. Waibel. Speaker-Independent Connected Letter Recognition With a Multi-Sate Time Delay Neural Network. In *3rd European Conf. on Speech, Communication and Technology*, Sept. 1993.

[6] H. Ney. The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. In *Transactions on Acoustics, Speech, and Signal Processing*, pages 263–271. IEEE, April 1984.

[7] H. Sakoe. Two Level DP-Matching – A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition. In *Transactions on Acoustic, Speech, Signal Processing*, volume ASSP-27, pages 588–595, Dec. 1979.

[8] M.Woszczyna, N.Aoki-Waibel, F.D.Buø, N.Coccaro, K.Horiguchi, T.Kemp, A.Lavie, A.McNair, T.Polzin, I.Rogina, C.P.Rose, T.Schultz, B.Suhm M.Tomita, A.Waibel Janus 93: Towards Spontaneous Speech Translation. In *Proc. International Conf. on Acoustics, Speech, and Signal Processing*. IEEE, May 1994.