N. Pavesic, H. Niemann, S. Kovacic, F. Mihelic (Eds.), *3rd Slovenian-German and 2nd SDRV Workshop on Speech and Image Understanding*, April 24-26, 1996, Ljubljana, Slovenia, pp. 105-114.

INTEGRATING LARGE CONTEXT LANGUAGE MODELS INTO A REAL TIME WORD RECOGNIZER

F. Gallwitz[1], E.G. Schukat-Talamazzini[2], H. Niemann[1]

[1] Lehrstuhl für Mustererkennung
Universität Erlangen–Nürnberg
Martensstr. 3
91058 Erlangen

[2] Institut für Informatik
Friedrich-Schiller-Universität Jena
Am Leutragraben 1
07740 Jena

Abstract

In this paper we present a new recognizer architecture that allows the efficient integration of language models with arbitrary large context information, e.g. polygram models, into the recognition process. Instead of using these models for rescoring the n best word chains generated using bigram information, we extract the best word chain, or optionally the n best word chains, directly from the word lattice using an A* algorithm that incorporates full language model information. For comparison, we developed an improved architecture for fast generation of the n best word chains using bigram information. Experimental results show, that direct incorporation of full language model information increases word accuracy significantly even when compared to rescoring the 1000 best word chains. At the same time, computation time is drastically reduced.

1 Introduction

It is well known that the consideration of language constraints is vital for effective and efficient speech recognition. Typically, these language constraints are modeled in a stochastic *language model* which will restrict the allowed sequences of words in an utterance [5]. The *a priori* probability $P(\boldsymbol{w})$ for a word sequence $\boldsymbol{w} = w_1 w_2 \ldots w_m$ can be expressed as a product of conditional probabilities $P(w_t|w_1 w_2 \ldots w_{t-1})$. Approximation of the *history* of the word w_t is done by limiting the number of considered preceding words to $N-1$. For this *N-gram* approach N is typically restricted to $N = 2$ (bigram) or $N = 3$ (trigram). *Polygram* language models (section 2) allow for a robust estimation of language model parameters for even larger values of N.

$$P(\boldsymbol{w}) \;=\; P(w_1) \cdot \prod_{t=2}^{m} P(w_t| \underbrace{w_{t-n+1} \ldots w_{t-1}}_{N-1}) \qquad (1)$$

Another type of stochastic language models are category based N-gram models [2]. Words are pooled in categories or word classes, usually under linguistic aspects. If one word is allowed to belong to more than one category, all possible category sequences $\boldsymbol{z} = z_1 z_2 \ldots z_m$ leading to a word sequence $\boldsymbol{w} = w_1 w_2 \ldots w_m$ have to be considered when calculating its probability:

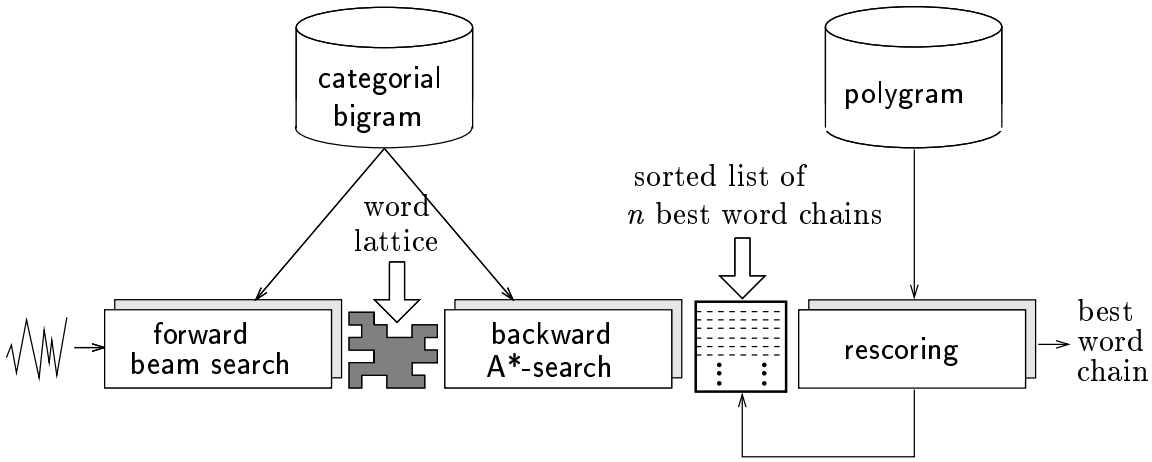


Figure 1: Recognizer architecture based on lattice-n-best algorithm and polygram rescoring of the n best word chains [6]

$$P(\boldsymbol{w}) = \sum_{\boldsymbol{z}} P(z_1)P(w_1|z_1) \cdot \tag{2}$$

$$\cdot \prod_{i=2}^{m} P(z_i|\underbrace{z_{i-N+1} \ldots z_{i-1}}_{N-1})P(w_i|z_i) \tag{3}$$

Unfortunately, the search space of a Viterbi continuous speech decoder grows exponentially with the order N of the language models. Thus, for large vocabulary real time Viterbi decoding on standard hardware, context is normally reduced to $N = 2$. On the other hand, language models with $N > 2$ usually have lower perplexities, i.e. they lead to better word accuracy.

One solution for integrating large context language models into the recognizer efficiently is the n-best-paradigm. The recognition process is separated into two steps: First, the n best word chains are generated using a bigram language model. These are rescored using a higher order language model, for example a polygram model, which can be done with very little computational cost.

There are different algorithms for generating the n best word chains [6]. One solution that causes little overhead is the *lattice-n-best* algorithm, which was first proposed in [13, 12]. It is only suboptimal, i.e. it does not always generate the n globally best word chains, but it is much faster than exact n-best algorithms [6]. The lattice-n-best algorithm is separated into a time synchronous forward decoding, in which a *word lattice* is generated, and a backward search for generating the n best word chains. The resulting recognizer architecture with polygram rescoring is shown in Figure 1.

In this paper, we present two modifications of the lattice-n-nbest algorithm that significantly improve its performance. In section 2, we explain some details of the polygram

language models. In section 3, we discuss details of the lattice-n-best algorithm and show some drawbacks of this approach. We will also present a modified architecture to generate the n best word chains more efficiently. In section 4 we show how we integrate polygram language models directly into the A* algorithm to avoid n-best rescoring completely. Experimental results are presented in section 5.

2 Polygram Language Models

In order to fully exploit the information present in the lower-order language models, the competing estimates can be combined by linear interpolation:

$$\tilde{P}(w|uv) \;=\; \varrho_0 \frac{1}{L} \;+\; \varrho_1 \hat{P}(w) \;+\; \varrho_2 \hat{P}(w|v) \;+\; \varrho_3 \hat{P}(w|uv) \tag{4}$$

where L is the size of the vocabulary $\mathcal{V}$. The weights ϱ_i of this convex combination of conditional trigram, bigram, unigram, and zerogram $(1/L)$ probabilities can be optimized with respect to a cross validation data set running the well known EM algorithm [1]. For a parametric refinement of the model, the ϱ_i may be considered functionally dependent on (part of) the word history uv. This approach can be generalized to the *polygram* formula introduced in [7]:

$$\tilde{P}(w|\boldsymbol{v}) \;=\; \sum_{i=0}^{N} \varrho_i \cdot \hat{P}_i(w|\boldsymbol{v}) \;, \quad \boldsymbol{v} \in \mathcal{V}^{N-1} \tag{5}$$

Thus, smoothed N-gram probabilities — of arbitrary order N — are obtained by combining all i-gram language models, $i = 0, \ldots, N$. The word predictor $\hat{P}_i(w|\boldsymbol{v})$ in eq. (5) denotes the maximum likelihood estimate of the conditional i-gram probability for word w emerging from context $\boldsymbol{v}$.

A non-linear smoothing process called *rational interpolation* for conditional N-gram probabilities was introduced in [10]:

$$\tilde{P}(w|\boldsymbol{v}) \;=\; \frac{\sum_{i=0}^{N} \varrho_i \cdot g_i(\boldsymbol{v}) \cdot \hat{P}_i(w|\boldsymbol{v})}{\sum_{i=0}^{N} \varrho_i \cdot g_i(\boldsymbol{v})} \;=:\; \frac{Q(w|\boldsymbol{v})}{Q(\boldsymbol{v})}, \tag{6}$$

where $g_i(\boldsymbol{v})$ is a history-dependent weight function. For details on language model interpolation strategies refer to [4, 10].

3 Modified Lattice-n-Best Algorithm

The basic idea of the lattice-n-best algorithm is to keep additional costs to generating only the best word chain as low as possible: During the forward beam search, a big number of different paths is generated, each representing a possible sequence of words. Instead of only generating the best scoring path at the end of the utterance, all words

that are included in one of the alternative search paths are stored as *word hypotheses* $I_{t_b}^{t_e}(w_j, z_l) := (w_j, z_l, t_b, t_e, v(w_{j,l}))$, where t_b and t_e are the first and last time frame of the word hypothesis, respectively, w_j is the word's lexicon index, z_l is the word category, and $v(w_{j,l})$ is the acoustic score of the word hypothesis. Of course, z_l does not have to be stored if only pairwise disjoint word categories are allowed. The result of the forward beam search is a set of alternative word hypotheses with $t_b \geq 1$ and $t_e \leq T$, where 1 is the first frame of the utterance and T is the last frame. This set of word hypotheses is called *word lattice*, and no path information is associated with the word hypotheses. Additionally, the maximum category-dependent path probabilities $\alpha_t(s_\tau(l))$ are stored for all categories z_l, with $1 \leq l \leq D$, where D is the number of categories and $s_\tau(l)$ is the symbolic end node of all word models in category z_l. These probabilities are used in the second step of the lattice-n-best algorithm, the backward A* search.

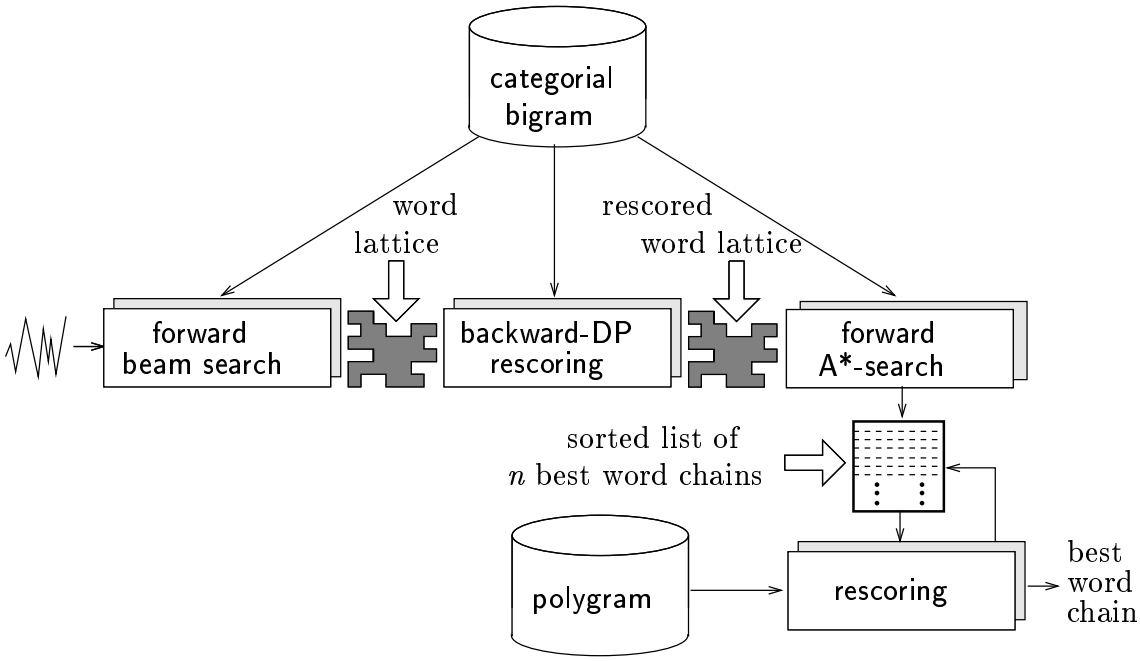
The A* algorithm is a *heuristic* approach to determining the *minimal cost path* or the n minimal cost paths in a finite state-space, details can be found in [8, 9]. The basic idea is that nodes on promising paths are *expanded* successively, meaning that an estimate $\hat{f}$ of the total path costs f is calculated for all possible successor nodes:

$$\hat{f}(v_i) \;=\; \hat{g}(v_i) + \hat{h}(v_i) \tag{7}$$

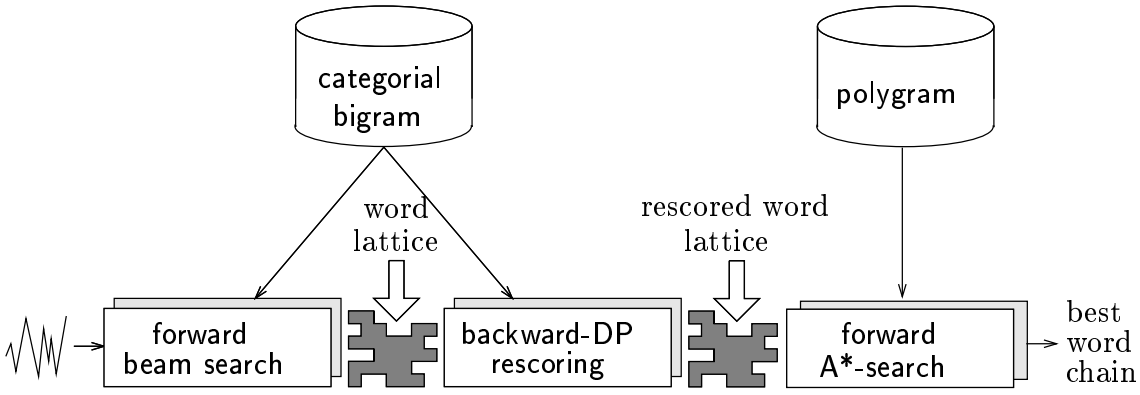
An estimate $\hat{g}(v_i)$ of the costs from the start node to the current node v_i can be calculated by summing up costs of all transformations along this path. The algorithm's efficiency depends on a good estimate of the remaining costs $\hat{h}(v_i)$ from the current node v_i to the goal. In the lattice-n-best algorithm we start the search at the end of the utterance and search for the minimal cost paths through the word lattice to the beginning of the utterance. Finding the path with the highest probability p is equal to finding the path with minimum costs $-\log(p)$, which gives us an additive cost function. In our case, $\hat{g}(v_i)$ is the sum of all acoustic costs and bigram costs from the end of the utterance to the current node, it can be calculated iteratively. The reason for searching in *backward* direction is, that we can now easily calculate an estimate of $\hat{h}(v_i)$ for each node, because during forward beam search, the maximum category dependent path probabilities $\alpha_t(s_\tau(l))$ were stored [6]:

$$\hat{h}(I_{t_{a_j}}^{t_{a_{j+1}}}(w_j, z_{l_j})) = \begin{cases} -\log(P_{anf}(z_l)^\vartheta) & \text{for} \quad t_{a_j} = 1 \\ -\log(\max\limits_{1 \leq k \leq D}\{\alpha_{t_{a_j}}(s_{\tau(k)}) \cdot B(z_k, z_{l_j})^\vartheta\} \cdot \varpi) & \text{for} \quad t_{a_j} > 1 \end{cases}, \tag{8}$$

where $B(z_l, z_k) = (P(z_l \mid z_k)P(w_l \mid z_l))$ is the bigram probability of word w_l when the predecessor category is z_k, and ϑ is the linguistic weight factor. Because the same bigram probabilities and linguistic weight factor were used during forward search, this is an *optimal* estimation of the remaining costs; it is identical with the costs of the best path to the beginning of the utterance. That means, that only nodes lying on the n optimal paths have to be expanded to generate the n best word chains. Thus, this algorithm is expected to be very fast compared to the computation time necessary for building the word lattice.



(a)



(b)

Figure 2: (a) Recognizer architecture based on forward beam search, lattice rescoring, bigram directed A* search, and polygram rescoring of the n best word chains (b) Recognizer architecture based on forward beam search, lattice rescoring, and polygram directed A* search

This is actually true for relatively short utterances. The number of alternative word hypotheses in the lattice is low at the beginning and end of the utterance, and its maximum is around the middle of the utterance. For short utterances of about 5 seconds, a typical value for the number of alternative word hypotheses ending in one frame is about 20, depending on the acoustic quality and the beam-width used during forward beam search. But when utterances are longer, about 30 seconds or more, this value may easily become two orders of magnitude higher than this. That means, that the computation time for generating 100 best, 1000 best or even more word chains is no more tolerable, because the time for expanding one single node grows linearly with the number of word hypotheses per time frame.

A heuristic solution of this problem is the restriction of word hypotheses per frame by defining a fixed maximum number m. Only the best scoring m words ending in one time frame are stored in the word lattice. This limits the computation time for expanding one node, but another problem arises: The cost estimation function $\hat{h}$ is no longer exact. This is due to the fact, that $\alpha_{t_{a_j}}(s_{\tau(k)})$ was calculated using words that possibly do *not* belong to the best m words ending in one frame. In consequence, we found it almost impossible to generate n-best word chains with the lattice-n-best algorithm for utterances longer than about 20 seconds. For many utterances, it was even practically impossible to find the single best word chain using this algorithm, because 100000 and more nodes were expanded without reaching the beginning of the utterance.

To solve this problem, we completely ignore the α-scores calculated during forward beam search. Instead, we rescore the lattice using dynamic programming. The rescoring is based on the categorical bigram and on the acoustic scores associated with the word hypotheses. If the rescoring is done in a forward direction and the number of word hypotheses per frame is not restricted, the new α'-scores and the α-scores do not differ. If the number of word hypotheses per frame is restricted to about $m = 20$, the time for rescoring the word lattice is almost negligible. But now the A* algorithm for the n-best generation is much faster and can handle long utterances without any problems, because now it is based on an *exact* estimation of the remaining costs. It generates word chains with increasing costs; the best word chain is always found first [9]. Of course, the search direction of the $A^\star$ algorithm is no more important, if the α-scores are recalculated. We can just as well do a backward rescoring of the word lattice and a forward $A^\star$ search. Both approaches produce identical results. The resulting architecture with polygram rescoring of the n best word chains is shown in Figure 2a.

4 Polygram directed A* Search

The generation of n best word chains in linear time still does not solve the main problem of the n-best paradigm: To ensure that a constant amount of all word chains in the word lattice is generated, n obviously has to grow *exponentially* with the number of words in an utterance. For example, if 3 alternative hypotheses exist for each word in an utterance of 15 words, $n = 3^{15} = 14,348,907$ different word chains could be generated.

This is the reason why we implemented another architecture that directly integrates

sample	calls	user turns	words
training	804	7732	27852
validation	54	441	1577
test	234	2383	8346

Table 1: Overview of training-, validation-, and test sample

polygram language models into the A* algorithm (Figure 2b). As before, we use bigram information for rescoring the word lattice with dynamic programming to get an estimate for the remaining costs $\hat{h}$, but with a slight difference: Instead of using the language model weight ϑ, which is optimized for best performance of the forward beam search, we use the polygram language model weight ϑ_p. This parameter can easily be optimized by rescoring the n best word chains of a validation sample, and its optimal value is usually significantly higher than the optimum value of ϑ. For the experiments described in section 5, the parameters were optimized to $\vartheta = 1.5$ and $\vartheta_p = 5.0$. We can use polygram language model information to score nodes during the A* search, because for every node v_i in the state space the complete history up to the beginning of the utterance can be determined (Figure 3). We can use this history to calculate a $\hat{g}$-score for all possible successor nodes using language model scores according to eq. (1) for arbitrary large values of N. But now our estimation of the remaining costs $\hat{h}$ is no longer exact. We can keep differences between polygram and bigram scores low by training both models on the same data, but $\hat{h}$ is no lower bound for the real path costs to the end of the utterance. Thus, we do not always find the best path *first*. Nevertheless, there is a good chance that the first path found using polygram information is at least as good as one determined by rescoring a large number of word chains that were generated using bigram information. We can additionally generate more than one word chain, which increases the probability of finding the globally best one. We will evaluate this in section 5.

5 Experiments and Results

As our recognizer is used as a module of our spoken dialog system [3, 11] which is able to answer inquiries about German Intercity train time tables. the results reported in this paper were evaluated using our corpus of spontaneous speech data collected by our system. It totals to 8 h 36 min. of speech and is divided into a training sample used for training of acoustic parameters and language models, a validation sample for optimizing recognizer parameters, and a test sample (Table 1).

The average number of words per utterance in the test sample is only 3.5, which should be an ideal domain for n-best rescoring. We trained both a polygram model ($N = 5$) and a bigram model using a system of 300 handcrafted word categories and rational interpolation (eq. (6)) on the training sample. We generated the 1000 best word chains using the modified lattice-n-best algorithm for all utterances in the validation test sample to optimize the polygram weight ϑ_p. Then we used the setup shown in Figure 2a

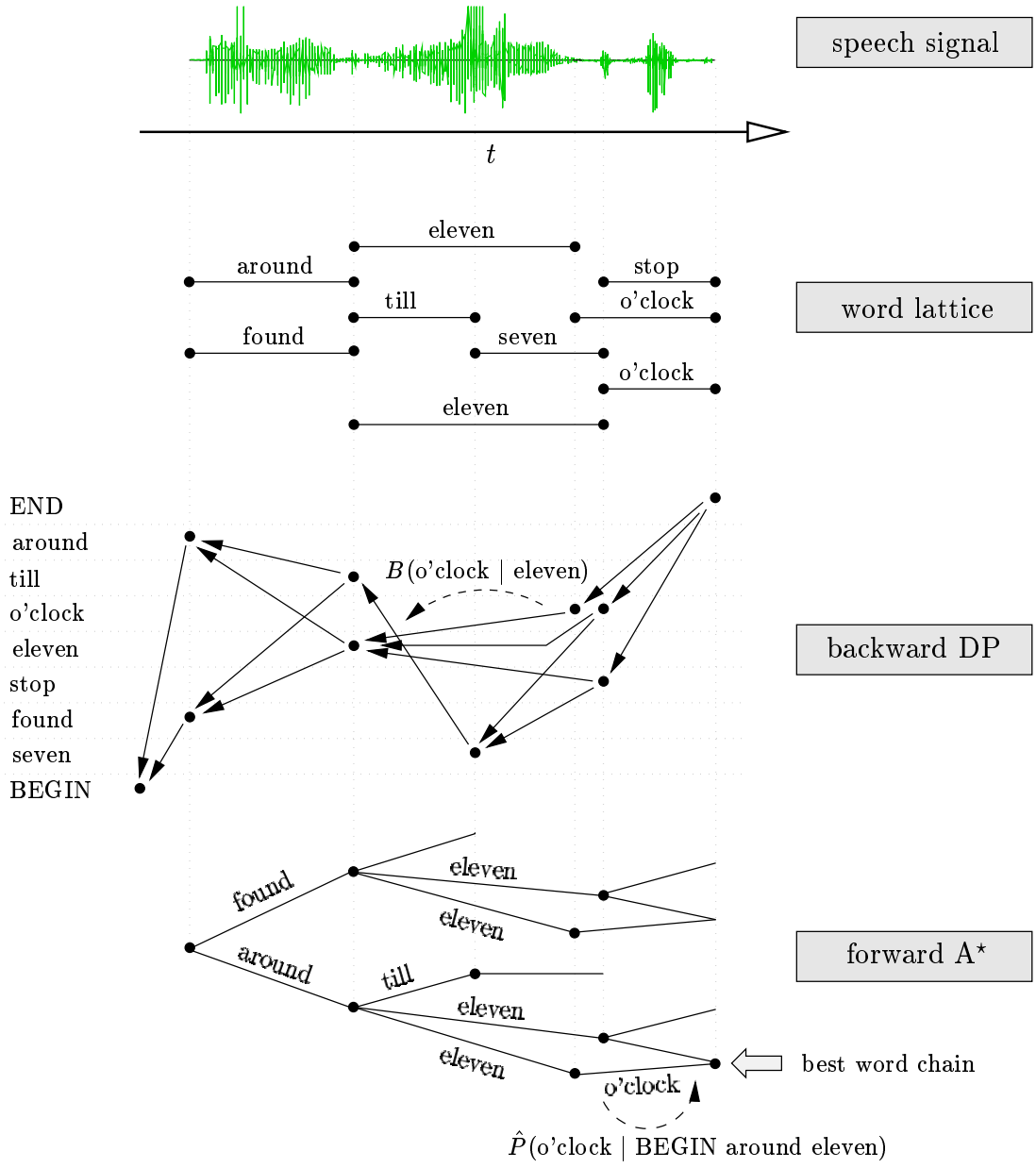


Figure 3: Word recognition based on lattice rescoring and polygram directed A* search. Bigram information is used for rescoring the word lattice, while polygram information is used for the forward A* search to extract the best word chain, or, alternatively, the n best word chains. Note that the state space searched by the A* algorithm is no tree, because due to word border shifts different paths representing identical word chains may end in one time frame, and thus may be recombined. If only the best word chain is required and polygram history is restricted to $N-1$ words, paths ending in one time frame with equal sequences of $N-1$ words may be recombined, too.

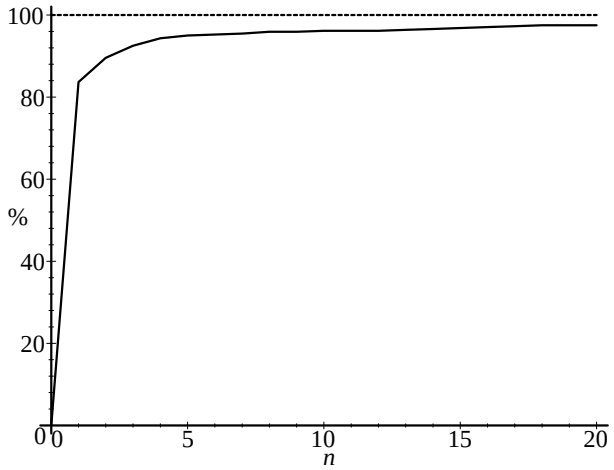


Figure 4: Amount of utterances where the globally best path in the word lattice is one of the first n word chains generated by polygram directed A* search

architecture	error rate	error red.	real time fac.
no rescoring	23.54 %	-	2.18
10-best rescoring	22.04 %	6.4 %	3.24
100-best rescoring	21.69 %	7.9 %	5.29
1000-best rescoring	21.68 %	7.9 %	7.08
polygram dir. A* (1-best)	20.94 %	11.0 %	2.44

Table 2: Comparison of word error rate and real time factor

with $n = 1000$ to generate the best word chains on the test sample. Using the same polygram weight, we generated only the first word chain using the setup from Figure 2a. Figure 4 shows, that generating more than $n = 1$ word chain with the polygram directed A* algorithm increases the probability of finding the globally best word chain. For values larger than $n = 5$ this increase is only very small, while computation time increases linearly with n.

The resulting word error rates and real time factors can be seen in Table 2. Polygram directed A* search is significantly faster than rescoring only the 10 best word chains, and it achieves a recognition accuracy that even outperforms 1000-best rescoring.

6 Conclusions

In this paper we presented different word recognizer architectures that use large context language model information to generate the best word chain. Our results show, that even on a sample of very short utterances (3.5 words average) n-best rescoring does not perform as well as a direct extraction of best word chains from the word lattice using the

polygram directed A* algorithm. We expect the algorithm to work equally well on longer utterances, where n-best rescoring cannot significantly reduce word error rate because of the exponentially increasing number of different word chains.

Acknowledgements

The work presented in this paper was partly supported by the DFG (German Research Foundation) under contract number 810 830-0. The authors would also like to sincerely thank Thomas Kuhn, now at Daimler-Benz AG, Ulm, for valuable suggestions and discussions.

References

[1] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–22, 1977.

[2] A.-M. Derouault and B. Merialdo. Natural Language Modeling for Phoneme-to-Text Transcription. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):742–749, 1986.

[3] W. Eckert, T. Kuhn, H. Niemann, S. Rieck, A. Scheuer, and E.G. Schukat-Talamazzini. A Spoken Dialogue System for German Intercity Train Timetable Inquiries. In *Proc. European Conf. on Speech Technology*, pages 1871–1874, Berlin, 1993.

[4] U. Essen and V. Steinbiss. Cooccurrence Smoothing for Stochastic Language Modeling. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 161–164, San Francisco, 1992.

[5] F. Jelinek, R.L. Mercer, and L.R. Bahl. Continuous Speech Recognition. In P.R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 549–573. North-Holland, 1982.

[6] T. Kuhn. *Die Erkennungsphase in einem Dialogsystem*, volume 80 of *Dissertationen zur Künstlichen Intelligenz*. infix, St. Augustin, 1995.

[7] T. Kuhn, H. Niemann, and E.G. Schukat-Talamazzini. Ergodic Hidden Markov Models and Polygrams for Language Modeling. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 357–360, Adelaide, Australia, 1994.

[8] N.J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Co., Palo Alto, CA, 1980.

[9] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA, 1984.

[10] E.G. Schukat-Talamazzini. Stochastic language models. In *Electrotechnical and Computer Science Conference*, Portoroz, Slovenia, 1995.

[11] E.G. Schukat-Talamazzini, T. Kuhn, and H. Niemann. Speech Recognition for Spoken Dialog Systems. In H. Niemann, R. De Mori, and G. Hahnrieder, editors, *Progress and Prospects of Speech Research and Technology*, number 1 in Proceedings in Artificial Intelligence, pages 110–120. Infix, 1994.

[12] R. Schwartz and S. Austin. Efficient, High-Performance Algorithms for N-Best Search. In *Speech and Natural Language Workshop*, pages 6–11, Hidden Valley, Pennsylvania, 1990. Morgan Kaufmann.

[13] V. Steinbiss. Sentence-Hypotheses Generation in a Continuous-Speech Recognition System. In *Proc. European Conf. on Speech Technology*, volume 2, pages 51–54, Paris, 1989.